Korpela Jaakko & Koski Samuel

”Muistan omilta kouluajoilta atk -kurssin ohjelmointihetken, jossa tunti opeteltiin kirjoittamaan pascal -koodia ja lopputulemana näytölle ilmestyi pikselin kokoinen piste.”

Tapaustutkimus ohjelmoinnin opettamisesta suomalaisissa peruskouluissa

Pro gradu -tutkielma
KASVATUSTIETEIDEN TIEDEKUNTA
Teknologiapainotteinen luokanopettajakoulutus
2021

Oulun yliopisto
Kasvatustieteiden tiedekunta
”Muistan omilta kouluajoilta atk -kurssin ohjelmointihetken, jossa tunti opeteltiin kirjoittamaan pascal -koodia ja lopputulemana näytölle ilmestyi pikselin kokoinen piste.” Tapaustutkimus ohjelmoinnin opettamisesta Suomen peruskouluissa. (Jaakko Korpela & Samuel Koski)
Pro gradu -tutkielma, 77 sivua, 7 liitesivua
Toukokuu 2021

Ohjelmoinnin ja ohjelmoinnillisen ajattelun nähdään kansainvälisesti lukeutuvan osaksi tulevaisuuden taitoja. Digitalisoituva yhteiskunta ja sen tarpeet ovat lähtökohtana näiden taitojen tärkeydelle. Ohjelmoinnin opetus sisällytettiin osaksi Perusopetuksen opetussuunnitelmien perusteita vuonna 2014. Tämä ei ole vielä kuitenkaan tarjonnut riittävän yhtenäisiä lähtökohtia ohjelmoinnin opetukselle kansallisella tasolla suomalaisissa peruskouluissa. Tässä pro gradu -tutkielmassa teoreettisen viitekehyksen muodostavat ohjelmointi, ohjelmoinnillinen ajattelu sekä näiden välinen yhteys.

Tutkielman tavoitteena oli selvittää, millaisin eri tavoin ohjelmointia opetetaan suomalaisissa peruskouluissa ja määritellä, mitä ohjelmoinnin opetus pitää sisällään. Ohjelmoinnin opetusta lähestyttiin ohjelmoinnillisen ajattelun kehityksen kautta, ja tutkielman päämääränä oli selvittää, minkälaisia käsitteitä, käytänteitä ja päämääriä opetus sisältää. Tutkielma toteutettiin laadullisena tapaustutkimuksena, jota analysoitiin teoriaohjaavan sisällönanalyysin keinoin. Tutkielman aineisto koostui 42 kyselytutkimukseen osallistuneen opettajan tai muun kasvatusalan ammattilaisen vastauksista.

Tutkielman tuloksissa selvisi, että suomalaisissa peruskouluissa tapahtuvassa ohjelmoinnin opetuksessa korostuvat keskeiset ohjelmoinnin käsitteet tukevat ja toisaalta tukeutuvat aiemmissa tutkimuksissa ja julkaisuissa mainittuihin keskeisiin ohjelmoinnin käsitteisiin. Tulokset myös osoittivat, että ohjelmointia opetetaan suomalaisissa peruskouluissa hyödyntäen useita eri ohjelmoinnin opetukselle teoriankin mukaan ominaisia työtapoja, toimintamalleja ja tehtävätyyppejä. Tulosten mukaan ohjelmoinnin opetuksen päämäärät suomalaisissa peruskouluissa niin ikään kulkevat käsikkäin teorian kanssa, jonka mukaan ohjelmoinnin opettamisen päämäärien tulisi sijaita itsensä ohjelmointitaidon ulkopuolisten, laaja-alaisempien tavoitteiden, kuten monilukutaidon, ajatteluntaitojen sekä työ- ja elinkeinoelämän tarpeiden ja vaateiden luona.

Avainsanat: ohjelmoinnin opetus, ohjelmointi, ohjelmoinnillinen ajattelu, uudet lukutaidot

Sisältö

1. Johdanto

Ohjelmoinnin opettaminen liitettiin osaksi perusopetusta vuonna 2016 käyttöönotetuissa Perusopetuksen opetussuunnitelman perusteissa. Se sisällytettiin osaksi matematiikan ja käsitöiden sisältöjä ja tavoitteita, sekä laaja-alaisten osaamistavoitteiden tieto- ja viestintäteknologista osaamista (L5). (Opetushallitus [OPH], 2014.) Ohjelmointi ja ohjelmoinnillinen ajattelu luetaan kansainvälisesti osaksi tulevaisuuden taitoja (*eng. 21st century skills)* (esim. International Society for Technology in Education [ISTE], 2021, Tsai, Wang & Hsu 2019, McClelland & Grata, 2018, Bocconi, Chioccariello, Dettori, Ferrari & Engelhardt, 2016). Digitalisoituva maailma ja siitä johtuva työmarkkinoiden muutos ovat lähtökohtina näiden taitojen tärkeydelle (Bocconi ym., 2016; McCormack, 2014). Koska tarve ohjelmoinnilliseen ajatteluun on lisääntynyt yhteiskunnallisella tasolla, on ohjelmoinnin opetuksen nykytilan ja sen tulevaisuuden tarkastelu tärkeää (Yadav, Good, Voogt & Fisser, 2017). Perusopetuksen opetussuunnitelman perusteiden 2014 sisältämä velvoite ohjelmoinnin opettamiseen ei ole vielä kuitenkaan tarjonnut riittäviä lähtökohtia yhtenäiselle ohjelmointiopetukselle maanlaajuisesti (OPH, 2021). Tästä syystä Opetus- ja kulttuuriministeriö on lanseerannut Uudet lukutaidot[1] -kehittämisohjelman osana laajempaa Oikeus oppia -kehittämisohjelmaa. Uudet lukutaidot -kehittämisohjelman voidaan nähdä laajentavan ja sanallistavan Opetushallituksen (2014) Perusopetuksen opetussuunnitelman sisältämiä sisältöjä ja tavoitteita ohjelmoinnista ja algoritmisesta ajattelusta.

Tutkielman kannalta keskeisiä aihepiirejä ovat ohjelmointi ja ohjelmoinnillinen ajattelu sekä niiden väliset yhteydet. Yksinkertaisimmillaan ohjelmoinnilla tarkoitetaan toimintaohjeiden antamista ennalta määrätyn toimenpiteen suorittamista varten (Hyvönen, Lappalainen & Lakanen, 2012). Mykkäsen ja Liukkaan (2014) kirjoittaman ohjelmoinnin oppaan *Koodi 2016* mukaan ohjelmointi on ohjeiden antamista ihmiseltä tietokoneelle. He näkevät ohjelmoinnin luovana ongelmanratkaisuna, jossa ongelma muutetaan pala palalta tietokoneelle ymmärrettävään muotoon. Ohjelmoinnillisella ajattelulla tarkoitetaan yleisesti erilaisia ajattelun taitoja ja -käytänteitä, joita käytetään erilaisissa ongelmanratkaisua vaativissa tehtävissä hyödyntäen erilaisia tietojenkäsittelytieteille ominaisia perusperiaatteita (esim. Denning & Tedre, 2019, 2016; Mohaghegh & McCauley, 2016; Curzon, Black, Meagher & McOwan, 2009; Wing, 2006). Ohjelmoinnin on nähty olevan yhteydessä ohjelmoinnillisen ajattelun ja sen osa-alueiden kehitykseen. (esim. Wei, Lin, Meng, Tan, Kong & Kinshuk, 2021; Fidai, Capraro & Capraro, 2020;

[1] https://uudetlukutaidot.fi/

Kert, Erkoç & Yeni, 2020; Lye & Koh, 2014). Ohjelmoinnillisen ajattelun on esitetty jakautuvan ohjelmoinnin yhteydessä kolmeen eri ulottuvuuteen, jotka ovat *käsitteet (eng. concepts)*, *käytänteet (eng. practices)* ja *päämäärät (eng. perspectives)* (Brennan & Resnick, 2012). Tutkielmassa ohjelmoinnin opetusta tarkastellaan näiden kolmen ulottuvuuden näkökulmista.

Perusopetuksen opetussuunnitelman perusteissa (OPH, 2014) ohjelmoimiseen tai ohjelmoinnillisen ajattelun kehittymisen sisällöt ja tavoitteet ovat melko vaikeasti tulkittavissa. Perusopetuksen opetussuunnitelman perusteiden (OPH, 2014) mukaan alakoulusta lähtien oppilaita tulisi perehdyttää ohjelmointiin. Vuorisen (2019) mukaan tällöin ohjelmoinnin opettaminen jää opettajille, joilla ei välttämättä ole sen opettamiseen riittävää motivaatiota ja edellytyksiä. Myös Taina (2015) esittää huolensa artikkelissaan *Matematiikkalehti Solmussa* seuraavasti:

> *”Opettajalla on suurin rooli opetuksen onnistumisessa. Mitä tapahtuu, kun opettaja saa opetettavaksi uuden alueen, jota hän ei hallitse, joka ei kiinnosta häntä ja jossa osa oppilaista tietää selvästi häntä enemmän? Opettaja ahdistuu ja oppilaat turhautuvat. Ahdistunut opettaja ei jaksa innostua oppilaiden oppimisesta, ja turhautunut oppilas viettää mieluummin aikaa pelaamalla tai sosiaalisessa mediassa kuin seuraamassa hänelle triviaalia opetusta. Pahimmassa tapauksessa ohjelmoinnin opetuksesta tulee välttämätön paha, joka ei hyödytä ketään ja vain vie tunteja muilta aineilta.” (Taina, 2015)*

Omat intressimme tutkielman aihepiiriä kohtaan ovat heränneet erinäisten koulutuksellisten ja työelämään liittyvien asioiden johdosta. Pedagogisen näkemyksemme mukaan lapsia tulee koulussa kasvattaa aktiivisiksi, mutta kriittisiksi toimijoiksi tulevaisuutta silmällä pitäen. Koemme ohjelmointitaidon ja ohjelmoinnillisen ajattelun taitojen lukeutuvan vahvasti tulevaisuuden taidoiksi, joten ohjelmoinnin opetuksen tutkiminen peruskoulukontekstissa tuntuu tärkeältä ja sitä kautta mielekkäältä. Tarve saada lisätietoa ja lisätä omaa ymmärrystä ohjelmoinnin opetuksen menneisyydestä, nykytilasta ja tulevaisuudesta peruskouluissa sai meidät tarttumaan ja perehtymään ohjelmoinnin opetukseen.

Tämän tutkielman päällimmäisenä tavoitteena on selvittää, millaisin eri tavoin ohjelmointia opetetaan suomalaisissa peruskouluissa. Tutkielmaa voidaan kuvata tapaustutkimukseksi, jossa on viitteitä monimenetelmällisyydestä. Tutkielman aineisto kerätään jakamalla linkki sähköiseen kyselylomakkeeseen. Tutkielman aineistoa analysoidaan teoriaohjaavan sisällönanalyysin keinoin. Tapaustutkimuksessa tutkimuksen kohde on usein jokin kohde tai ilmiö, minkä avulla pyritään saamaan tarkkapiirteinen kuvaus tutkittavasta ilmiöstä (Laine, Bamberg & Jokinen,

2007). Monimenetelmällisyydellä (*eng. mixed methods)* luodaan parempaa kuvaa tutkimusongelmiin hyödyntäen sekä laadullista että määrällistä aineistoa (Tuomi & Sarajärvi, 2018).

Luvussa kaksi esitellään tutkimuksen tavoite ja tutkimuskysymykset. Tutkielman teoreettinen viitekehys esitellään luvussa kolme. Ohjelmoinnin opetuksen teoriaa suomalaisen peruskoulun kontekstissa esitellään aikaisemman tutkimustiedon perusteella luvussa neljä. Luku viisi sisältää kuvauksen tutkielman toteutuksesta, jossa kerrotaan tutkimuksen taustoista, vastaajajoukosta, aineiston keruusta ja analyysistä. Kuudennessa luvussa esitellään tutkimuksen tulokset ja lopuksi luku seitsemän sisältää tutkielman päätelmät ja yhteenvedon tutkielmasta.

2. Tutkimuksen tavoite ja tutkimuskysymykset

Ohjelmoinnin opettamista ei ole aiemmin tutkittu Suomessa vastaavalla tavalla tarkastellen ohjelmoinnin opetuksen käsitteitä, käytänteitä ja päämääriä, joten tutkimukselle on olemassa selkeä tarve ja sen tekeminen koettiin aiheelliseksi. Tutkielmassa selvitetään, millaisin eri tavoin ohjelmointia opetetaan peruskouluissa käytännön tasolla. Alla esiteltyihin tutkimuskysymyksiin etsitään vastauksia avoimia kysymyksiä sisältävällä kyselomakkeella. Tutkielmamme tavoitteena on selvittää, millaisin eri tavoin ohjelmointia opetetaan suomalaisissa peruskouluissa. Tutkielmamme avulla halutaan selvittää ja määritellä, mitä ohjelmoinnin opetus pitää sisällään – minkälaisia käsitteitä, käytänteitä ja päämääriä opetus sisältää.

Tutkimuskysymyksemme ovat:

1. Mitkä ohjelmointiin liittyvät käsitteet korostuvat ohjelmoinnin opetuksessa?

2. Millaisten tehtävien ja työtapojen avulla ohjelmointia opetetaan?

3. Millaisia näkemyksiä opettajilla on ohjelmoinnin opetuksen keskeisistä tavoitteista ja vaikutuksista oppilaiden tulevaisuudelle?

3. Ohjelmointi ja ohjelmoinnillinen ajattelu

Ohjelmointi *(eng. programming)* ja ohjelmoinnillinen ajattelu *(eng. computational thinking)* muistuttavat suomen kielessä toisiaan, mutta niiden merkitykset ovat erilaisia. Ohjelmoinnillinen ajattelu yhdistetään usein ohjelmointiin ohjelmoinnin toimiessa yhtenä ohjelmoinnillisen ajattelun oppimisen keinona ja käsitteiden konkretisoijana. Yleinen konsensus kuitenkin on, että ohjelmoinnillinen ajattelu on käsitteenä paljon laajempi, kuin pelkkä ohjelmointi. (Bocconi ym., 2016.) Useimmiten peruskouluissa ohjelmointi nähdään koodauksena ja ohjelmoinnillisen ajattelun olevan ”enemmän kuin koodausta” (Lye & Koh 2014; Wing 2006). Tässä luvussa käsitellään ohjelmoinnin ja ohjelmoinnillisen ajattelun käsitteitä peruskoulukontekstissa, miten ne tässä tutkielmassa määritellään. Ensimmäisessä alaluvussa käsitellään ohjelmoinnin määritelmää peruskoulukontekstissa, toisessa alaluvussa käsitellään ohjelmoinnillisen ajattelun määritelmää ja kolmannessa alaluvussa käsitellään näiden välistä yhteyttä.

3.1 Ohjelmointi peruskoulukontekstissa

Ohjelmoinnin määrittely suomalaisen peruskoulun kontekstissa on melko hankalaa, sillä nykyisellään käytössä olevassa perusopetuksen opetussuunnitelmassa (2014) ohjelmoinnin opetuksesta puhutaan yleisellä tasolla ”ohjelmointina”. Tässä luvussa ohjelmointi määritellään lyhyesti kasvatuksellisista näkökulmista siten, miten se myös suomalaisissa peruskouluissa, ja tässä tutkielmassa, nähdään. Tämän tutkielman teoreettisessa viitekehyksessä ei käsitellä ohjelmoinnin historiallista kehitystä, vaan nykytilannetta – sitä miten ohjelmointi suomalaisissa peruskouluissa määritellään osana tulevaisuuden taitoja *(eng. 21th century skills)*. Ohjelmoimisella ja koodaamisella *(eng. programming and coding)* tarkoitetaan useimmiten samoja toimintoja, jossa tietokoneelle annetaan ohjeita jonkin tehtävän suorittamiseksi (Balanskat & Engelhardt, 2015). Tässä tutkielmassa käytämme käsitettä ohjelmoiminen, joka kattaa myös koodaamisen käsitteen.

Ohjelmointia tai ohjelmoinnin kaltaista toimintaa esiintyy jokaisen ihmisen arkielämässä lähes päivittäin. Yksinkertaisimmillaan ohjelmointi on toimintaohjeiden antamista ennalta määrätyn toimenpiteen suorittamista varten. Alkeellista ohjelmointia on esimerkiksi mikroaaltouunin käyttäminen, sillä tällöin ihminen antaa uunille ohjeet vaikkapa ruoan lämmitykseen. (Hyvö-

nen ym. 2012.) Mykkäsen ja Liukkaan (2014) mukaan lyhyesti määriteltynä ohjelmointi on ohjeiden antamista ihmiseltä tietokoneelle. Heidän mukaansa nykyaikainen ohjelmointi nähdään luovana ongelmanratkaisuna, jossa ohjelmoijan tehtävä on ratkaista ongelma muuttamalla se ”pala palalta muotoon, jonka tietokone ymmärtää”. Myös Majaranta (2002) näkee ohjelmoinnin lähtökohdan olevan ongelmalähtöinen – tarve saada haluttu tehtävä suoritettua.

Peruskoulukontekstissa ohjelmointia ja koodien luomista ei pidä nähdä päämääränä tai itsetarkoituksena, vaan pikemminkin välineenä, jonka avulla voidaan oppia uusia ajattelun tapoja (Kangas & Vartiainen 2019). Ohjelmointi on tärkeä osa tieto- ja viestintäteknologiaa, sillä ohjelmointikielen osaaminen antaa mahdollisuuden ilmaista itseään tai ajatuksiaan tietoteknologisessa ympäristössä (Resnick ym. 2009). Kangas ja Vartiainen (2019) toteavat teoksessaan *Ohjelmoinnin ABC varhaiskasvatuksessa* ohjelmoinnin oppimisen kehittävän myös erilaisia ajattelun taitoja, kuten matemaatis-loogista ajattelua, luovaa ajattelua, ongelmanratkaisutaitokeskeistä ajattelua sekä leikillistä ajattelua. Majaranta (2002) toteaa ohjelmoinnin kehittävän ajattelun taitojen lisäksi myös analysoinnin taitoja.

Ohjelmointiin ja sen opettamiseen liittyy paljon erilaisia ohjelmointikieliä ja -alustoja. Kuten jokaisella kielellä, myös ohjelmointikielillä on omat sanastonsa ja kielioppisääntönsä, joita noudattamalla kieltä pystytään ymmärtämään ja käyttämään (Mykkänen & Liukas 2014). Ohjelmointikielet ovat tarkasti määriteltyjä, ja niiden toiminta perustuu syvästi logiikkaan, eikä niissä voi käyttää itse keksittyjä komentoja (Majaranta 2002). Tietyissä tapauksissa oppilas voi kuitenkin ohjelmoida eli syöttää komentoja laitteeseen myös ilman teoreettista tietoa ohjelmointikielistä (Otterborn, Schönborn & Hultén, 2019).

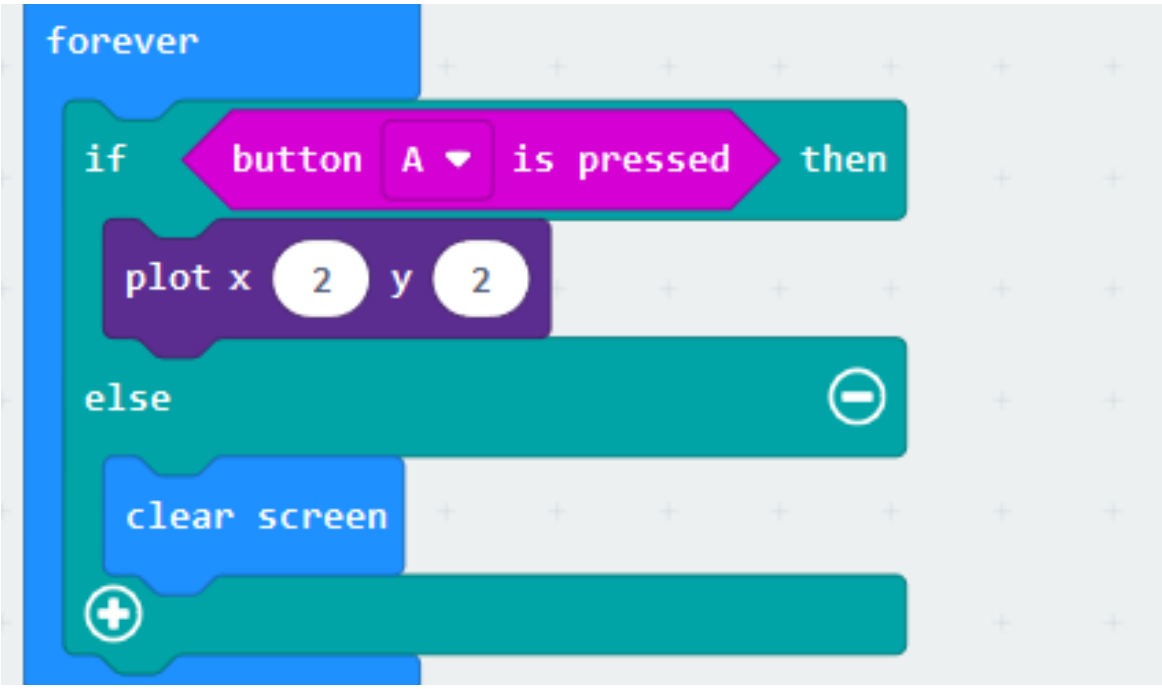
Ohjelmointiin liittyy paljon käsitteitä, jotka ovat yleisiä monilla ohjelmointikielillä. Ohjelmoinnin oppimisen käsitteitä on tutkittu vain vähän (ks. Moilanen 2018), mistä syystä näiden käsitteiden määritteleminen yksiselitteisesti on vaikeaa. Linda Liukas ja Juhani Mykkänen (2014) kirjoittivat Koodi 2016 -oppaan, joka kirjottajien omien sanojen mukaan on ensiapupakkaus ohjelmoinnista kiinnostuneille. Tämä opas sisältää konkreettisten ohjeiden lisäksi myös lyhyen tietopankin ohjelmoinnista ja sen opettamisesta. Keskeisimpänä ohjelmointisanastona esimerkiksi tässä oppaassa oli nostettu esiin käsitteet *tietokone, muuttuja, algoritmi, funktiot, lista, toistorakenne* ja *ehtolause*.

Ohjelmointi ja sen opettaminen vaatii myös valitun ohjelmointikielen lisäksi sille sopivan ohjelmointiympäristön. Jokaiselle ohjelmointikielelle voidaan ajatella olevan oma ohjelmointiympäristönsä, vaikka näiden ympäristöjen elementit muistuttavatkin toisiaan. Yksinkertaisimmillaan ohjelmointiympäristössä on vain tekstieditori, johon lähdekoodia kirjoitetaan (ks. kuvio 1). Koodi 2016 -oppaan mukaan alakoulussa ohjelmoinnin työkaluna ei käytetä varsinaista ohjelmointikieltä vaan visuaalista ohjelmointiympäristöä, jossa ohjelmoidaan hiiren avulla kirjoittamisen sijaan. Peruskoulun opetussuunnitelman (2014) mukaan kuudennen luokan lopussa hyvän osaamisen tason mukaan oppilas osaa ohjelmoida toimivan ohjelman graafisessa ohjelmointiympäristössä. Graafisella tai visuaalisella ohjelmointiympäristöllä tarkoitetaan ympäristöä, joka hyödyntää ohjelmoimisen apuna graafisia ja visuaalisia elementtejä kuten tekstiä, symboleita tai kuvia (ks. kuvio 2) (esim. Resnick ym., 2009). Ohjelmointiympäristö ei itsessään eroa merkittävästi muistakaan oppimisympäristöistä. Hyvä ohjelmointiympäristö, kuten muutkin oppimisympäristöt, tukee ja jäsentää oppijan oppimisprosesseja yksilöllisellä tasolla (Jalkanen, Järvenoja & Litola, 2012).

KUVIO 1. Arduino C -ohjelmointiympäristössä kirjoitettua koodia

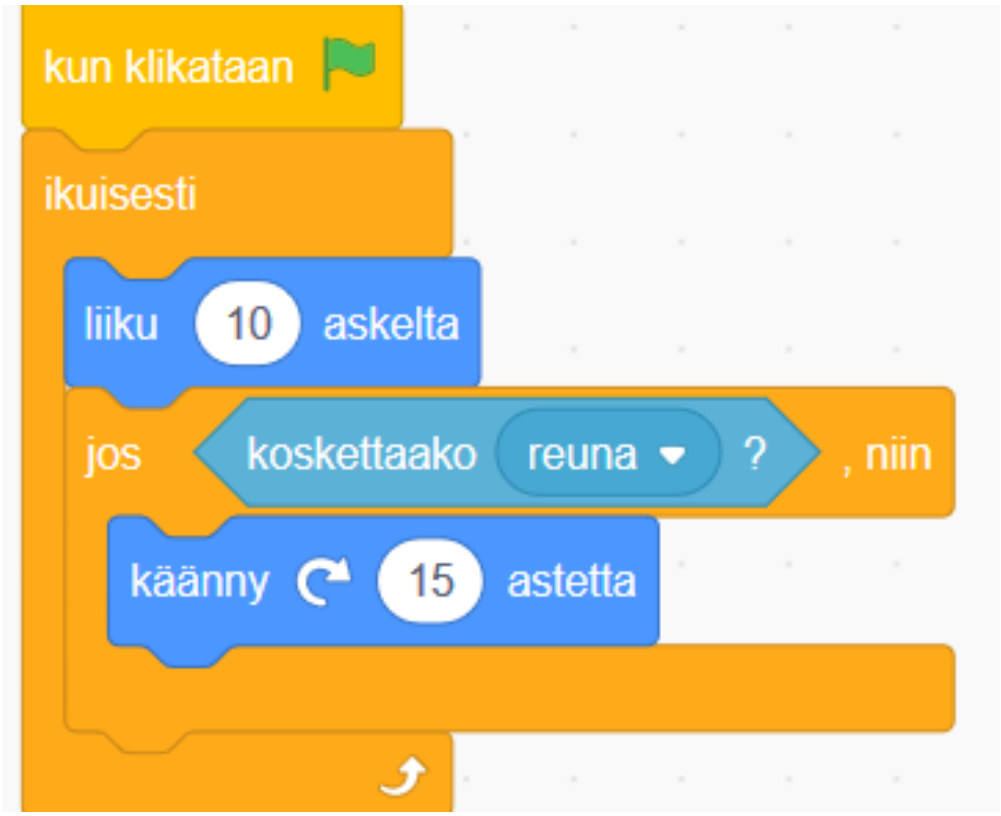
Ohjelmoinnin opetuksen onnistumisen kannalta, ja sen, että edesauttaisi oppijan ymmärryksen kehitystä, täytyy sen lähtökohtana olla ohjelmoinnilliseen ajatteluun opettaminen (Ala-Mutka, 2004). Koska tekstipohjaisten ohjelmointikielien suurimmaksi ongelmaksi oppilaiden kohdalla on havaittu koodien järjestelemisen hahmottaminen (Özmen & Altun, 2014), on tätä ongelmaa pyritty lähestymään kehittämällä visuaalisia ohjelmointikieliä ja -ympäristöjä, jotka helpottavat aloittelevan ohjelmoijan oppimista (Schwartz, Stagner & Morrison, 2006). On myös havaittu,

että visuaaliset ohjelmointiympäristöt kiihdyttävät oppijan ymmärryksen syntymistä (Naharro-Berrocal, Pareja-Flores, Urquiza-Fuentes & Velazques-Iturbide, 2002) sekä lisäävät oppilaiden motivaatiota ja halukkuutta oppimiseen (Sáez-López, Gonzáles & Cano, 2016).



KUVIO 2. Microsoft MakeCode lohko-ohjelmointiympäristössä tuotettua koodia

Visuaalisissa ohjelmointiympäristöissä ja -kielissä käytetään koodilohkoja (ks. kuvio 3), joiden tarkoitus on sekä helpottaa itse koodin rakentamista että ymmärryksen syntyä (Wilson & Meffat, 2010). Suosittuja alkeisohjelmoinnin opetuksessa käytettyjä visuaalisia ohjelmointiympäristöjä ja -kieliä ovat esimerkiksi Scratch (Malan & Leitner, 2007; Wu, Chang & He, 2010), Kodu (Stolee & Fristoe, 2011) ja StarLogo (Klopfer & Yoon, 2005). Mainitut visuaaliset ohjelmointiympäristöt tarjoavat ohjelmoinnin yhteydessä välittömän visuaalisen vastikkeen ohjelmoidulle koodille. Suomen Perusopetuksen opetussuunnitelman perusteiden (OPH, 2014) mukaan vuosiluokkien 3.–6. aikana ohjelmointia harjoitellaankin käyttäen visuaalisia (graafisia) ohjelmointiympäristöjä.



KUVIO 3. Scratch -ohjelmointiympäristössä tuotettua koodia koodilohkojen avulla

3.2 Ohjelmoinnillinen ajattelu

Termistä *ohjelmoinnillinen ajattelu (eng. computational thinking)* on muotoutunut viime vuosien aikana eräänlainen muotisana, jolla on useita toisistaan hieman eriäviä määritelmiä ja jopa poissulkevia määritelmiä. (Denning & Tedre, 2019; Mohaghegh & McCauley, 2016) Osittain tämän voidaan nähdä johtuvan termin havainnoidun historian varhaisesta vaiheesta, joka alkoi vuonna 2006 Jeannette Wingin julkaiseman artikkelin *Computational Thinking* myötävaikutuksesta. (Mohaghegh & McCauley, 2016; Wing, 2006) Yleisesti ottaen sillä tarkoitetaan kuitenkin kokoelmaa erilaisia ajattelun taitoja ja -käytänteitä, joita hyödynnetään erilaisiin ongelmanratkaisua vaativiin tehtäviin hyödyntäen erilaisia tietojenkäsittelytieteille ominaisia perusperiaatteita (esim. Denning & Tedre, 2019, 2016; Mohaghegh & McCauley, 2016; Curzon, Black, Meagher & McOwan, 2009; Wing, 2006). Se koostuu erilaisista ajattelun prosesseista, kuten loogisesta- ja algoritmisesta ajattelusta ja järjestelmien ymmärtämisestä (Tsai, Wang & Hsu 2019; Wing 2006). Joskus suomenkielisissä teksteissä ohjelmoinnillisesta ajattelusta käytetään

myös termejä *laskennallinen ajattelu, automatisointiajattelu* tai *algoritminen ajattelu* (Kekäläinen, 2015) ja esimerkiksi nykyisessä Perusopetuksen opetussuunnitelman perusteissa (OPH, 2014) siitä käytetään jälkimmäisenä mainittua termiä algoritminen ajattelu. Ohjelmoinnillisen ajattelun voidaan kuitenkin nähdä sisältävän algoritmista ajattelua laajemman käsityksen samasta aihepiiristä ja algoritmisen ajattelun lukeutuvan useissa tulkinnoissa yhdeksi ohjelmoinnillisen ajattelun osa-alueeksi (Mohaghegh & McCauley, 2016; Bocconi ym., 2016). Lisäksi Uudet lukutaidot -kehittämisohjelman (OPH, 2021) osaamisen kuvauksissa ohjelmointiosaamisen kohdalla puhutaan algoritmisen ajattelun sijaan jo ohjelmoinnillisesta ajattelusta. Tässä tutkielmassa *computational thinking* termin suomalaiseksi käännökseksi on siis näillä perustein valittu termi ohjelmoinnillinen ajattelu.

Ohjelmoinnillisessa ajattelussa ajattelun taitoja ja -käytänteitä hyödyntäen voidaan luoda komputaatioita, eli laskelmia, joita esimerkiksi tietokone osaisi tulkita suorittaakseen erilaisia tehtäviä, sekä kykyä selittää ja tulkita maailmaa monimutkaisina tietoprosesseina. (Denning & Tedre, 2019; Bocconi ym., 2016) Komputaatioita hyödynnetään maailmassa lähes kaikkialla erilaisten tehtävien suorittamiseen, kuten simulointiin siitä, miten lentokoneen siivet tulisi muotoilla liidon mahdollistamiseksi, miltä huomisen sää näyttää tai ketkä kaksi ihmistä muodostaisivat hyvän parin keskenään (Denning & Tedre, 2019). Komputaatiot voidaan ymmärtää myös data-analytiikkana tai dataan kohdistuvina laskelmina (National Research Council, 2013).

Ohjelmoinnillisen ajattelun nähdään lukeutuvan tulevaisuuden taitoihin (esim. ISTE, 2021; McClelland & Grata, 2018; Mohaghegh & McCauley 2016). Ohjelmoinnillisen ajattelun taitoa pidetäänkin joskus yhtä tärkeänä taitona, kuin esimerkiksi luku-, kirjoitus- ja laskutaitoa. (Tsai, Wang & Hsu 2019; Bocconi ym., 2016; Wing 2006) Myös kykyä ohjelmoida pidetään tärkeänä 2000-luvun taitona (Bocconi ym., 2016) ja sitä onkin verrattu kykyyn kirjoittaa; ”*People do not only program for solving problems,but also program for learning. Programming is like writing, and consequently,not only computer scientists should learn to program, just like not only novelists should learn to write*” (Segredo, Miranda, León & Santos, 2016, s. 11). Näiden taitojen tärkeyttä 2000-luvun maailmassa perustellaan muun muassa digitalisoituvalla maailmalla ja tästä johtuvalla työmarkkinoiden digitalisoitumisella. (Bocconi ym., 2016; McCormack, 2014) Useissa maissa kyky ohjelmoinnilliseen ajatteluun nähdään tärkeänä vaikuttajana yksilön mahdollisuuksiin toimia, osallistua ja innovoida digitalisoituvassa yhteiskunnassa (Bocconi ym., 2016, The White House, 2016).

3.3 Ohjelmoinnin ja ohjelmoinnillisen ajattelun yhteys

Ohjelmoinnilla ja koodaamisella on havaittu olevan yhteyksiä ohjelmoinnillisen ajattelun ja sen osa-alueiden kehitykseen. (esim. Wei, Lin, Meng, Tan, Kong & Kinshuk, 2021; Fidai, Capraro & Capraro, 2020; Kert, Erkoç & Yeni, 2020; Lye & Koh, 2014; Wing, 2006) Ohjelmointitaidon harjoittelu pelkän koodaamisen harjoittelun avulla on kuitenkin nähty riittämättömäksi, sillä pelkästä koodaustaidosta on vaikea päätellä, mitä oppija on oppinut, eikä koodaustaito sellaisenaan ole siirrettävissä muihin aihepiireihin (Kafai & Burke, 2013).

Koodauksen voidaan ajatella perustuvan taitoon ohjelmoinnista, joka koostuu algoritmeista ja niiden ymmärryksestä. Näiden taitojen kokonaisvaltainen hallinta vaatii ongelmanratkaisutaitoja, jotka ovat oppijan kannalta tärkeitä elämän perustaitoja ja jotka lukeutuvat osaksi ohjelmoinnillisen ajattelun taitoja. (Yucel, Karahoca & Karahoca, 2016; Trilling & Fadel, 2009.) Ohjelmointi tarjoaa kuitenkin kontekstin ohjelmoinnillisen ajattelun taitojen harjoittelulle (esim. Grover, Jackiw & Lundh, 2019; Brennan & Resnick, 2012). Ohjelmoinnillinen ajattelu puolestaan sisältää erilaisia ohjelmoinnissa ja koodaamisessa tarvittavia ajattelun taitoja ja käytänteitä, joita ovat aihepiiriä käsittelevän kirjallisuuden ja tieteellisten julkaisujen perusteella abstrahointi, algoritminen ajattelu, automaatio, hajottaminen tai erittely, virheiden jäljitys ja korjaaminen sekä kaavojen tunnistaminen (Bocconi ym., 2016).

Ohjelmoinnillisen ajattelun käyttö korostuu erityisesti tietokoneohjelmoinnin saralla – kuinka saada tietotekninen laite suorittamaan haluttu tehtävä halutulla tavalla. Ohjelmoinnillisen ajattelun avulla pyritään siis luomaan ja määrittämään sopivat algoritmit, joiden avulla voidaan yksityiskohtaisesti kertoa tietokoneelle, kuinka sen halutaan menettelevän halutun tehtävän suorittamiseksi. Laajemmalti tarkasteltuna ohjelmoinnillisen ajattelun avulla pyritään kuitenkin ymmärtämään myös ympäröivää maailmaa, jossa tietokoneohjelmat toimivat eräänlaisena siltana ympäröivän maailman ja ihmiselle syntyvän ymmärryksen välillä. Tietokoneohjelmien luomiseksi niiden tekijän täytyy ymmärtää itse tietokoneen toiminnan lisäksi myös tietokoneohjelman käyttäjiä ja sitä, miten tai mihin tarkoitukseen käyttäjä käyttää tai haluaisi käyttää tietokoneohjelmaa. (Denning & Tedre, 2019.)

Tietokoneohjelmat koostuvat algoritmeista, jotka on määritelty tiettyä tarkoitusta varten, ja ne tuottavat tietoa tai auttavat tulkitsemaan niihin syötettyä dataa. Esimerkiksi erilaiset tietokoneen avulla toteutetut simulaatiot auttavat ymmärtämään maailmaa tavalla, joka olisi käytännössä

mahdotonta ilman tietokoneesta saatavaa laskentatehoa. Ohjelmoinnillinen ajattelu auttaa myös ymmärtämään tietokoneiden laskennallisia rajoituksia – ymmärrys mahdottomista algoritmeista, puutteellisesta datasta tai laskentatehon riittämättömyydestä auttavat välttämään tietokoneen käyttämistä sellaisiin asioihin, joihin se ei sovellu. (Denning & Tedre, 2019.)

Ohjelmoinnillisen ajattelun on esitetty jakautuvan ohjelmoinnin yhteydessä kolmeen eri ulottuvuuteen, jotka ovat *käsitteet (eng. concepts)*, *käytänteet (eng. practices)* ja *näkökulmat (eng. perspectives)* (Brennan & Resnick, 2012). Ajatuksen ohjelmoinnillisen ajattelun jakamisesta kolmeen ulottuvuuteen esittelivät Brennan & Resnick (2012) tutkiessaan lasten ohjelmointioppimisen vaikutuksia ohjelmoinnillisen ajattelun kehitykseen visuaalisen ohjelmointiympäristö Scratchin käytön yhteydessä.

Ulottuvuusjaon on esitetty auttavan ohjelmoinnillisen ajattelun rakentumisen ymmärtämisessä ja sen kehityksen arvioinnissa ja havainnoinnissa (Brennan & Resnick, 2012), vaikkakin sen on myönnetty myös sisältävän omat haasteensa esimerkiksi sen suhteen, miten näiden kehitystä käytännössä voidaan mitata (Falloon, 2015; Brennan & Resnick, 2012). Osa-aluejaon on kuitenkin nähty avaavan peruskouluikäisten oppilaiden lähestymistapoja ohjelmointiin sekä käsityksiä ohjelmoinnista ja täten mahdollistavan niiden tulkintaa (Segredo, Miranda, León & Santos, 2016; Lye & Koh, 2014; Brennan & Resnick, 2012). Lisäksi sitä on käytetty aikaisemmissakin ohjelmoinnillisen ajattelun kehittymistä tutkivissa tutkimuksissa onnistuneesti tuloksia antavia mittareita käyttäen (esim. Mouza, Pan, Yang, & Pollock, 2020; Lye & Koh, 2014).

Ohjelmoinnillisilla käsitteillä tarkoitetaan käsitteitä, jotka ohjelmoija oppii ja käyttää ohjelmoidessaan (Lye & Koh, 2014; Brennan & Resnick, 2012). Tutkimuksessaan Brennan ja Resnick (2012) huomasivat muutamien ohjelmointiin liittyvien käsitteiden korostuvan lasten ohjelmointitehtävissä ja niiden olevan yleishyödyllisiä useassa eri Scratch-projektissa sekä muissa ohjelmointiin liittyvissä konteksteissa. Näitä käsitteitä olivat järjestys *(eng. sequences)*, toistot *(eng. loops)*, rinnakkaisuus tai yhtäaikaisuus *(eng. parallelism)*, tapahtumat *(eng. events)*, ehdot *(eng. conditionals)*, operaattorit *(eng. operators)* ja data tai tieto *(eng. data)*.

Ohjelmoinnillisilla käytänteillä tarkoitetaan ohjelmoijien tapoja työstää omia projektejaan. Kun käsitteiden tarkastelu ohjelmoinnillisen ajattelun kehityksessä antoi vastauksia siihen, mitä oppilas oppii, antaa käytänteiden tarkastelu puolestaan vastauksia siihen, miten oppilas oppii.

Nämä käytänteet ovat erilaisia ongelmanratkaisun työtapoja, jotka ilmenevät ohjelmointiprosessin aikana (Lye & Koh, 2014). Lasten keskuudessa esiintyviksi ohjelmoinnin käytänteiksi Brennan ja Resnick (2012) havainnoivat esimeriksi *kokeilemisen ja iteroinnin (eng. experimenting and iterating)*, *testaamisen ja virheiden korjaamisen (eng. testing and debugging), uudelleenkäytön ja uudelleenversioinnin (eng. reusing and remixing)* sekä *abstrahoinnin ja moduloinnin (eng. abstracting and modularizing)*. Tutkimuksessaan Brennan ja Resnick (2012) tekivät huomiot pohjautuen lasten kanssa käytyihin keskusteluihin, joissa lapset kuvasivat ymmärrystään omasta kehityksestään.

Ohjelmoinnillisilla näkökulmilla tarkoitetaan ohjelmoijan näkemyksiä omasta toimijuudestaan ja yhteydestään ympäröivään maailmaan ja muihin toimijoihin, eli sitä, miksi tai millä tarkoitusperillä ohjelmoi. Tällaisiksi ohjelmoinnin oppimisen näkökulmiksi esitetään itseilmaisu *(eng. expressing),* yhteisöllisyys *(eng. connecting)* sekä kyseenalaistaminen *(eng. questioning)*. (Brennan & Resnick, 2012.) Tässä pro gradu -tutkielmassa ohjelmoinnillisista näkökulmista puhutaan tulevaisuuslähtöisesti ohjelmoinnillisina päämäärinä.

Tätä ohjelmoinnillisen ajattelun jakoa kolmeen ulottuvuuteen hyödynnetään myös tämän tutkimuksen aineistonkeruussa. Vaikka Perusopetuksen opetussuunnitelmien perusteissa (OPH, 2014) ohjelmoinnillisesta ajattelusta puhutaankin termillä algoritminen ajattelu, voidaan sen nähdä viittaavan silti suurempaan kokonaisuuteen. Ulottuvuusjaon hyödyntäminen aineistonkeruussa on perusteltua suomen peruskouluissa tapahtuvan ohjelmoinnin opetuksen pääpainopisteen ollessa ajattelun taitojen kehityksessä, (OPH, 2014) ja tutkielman pyrkiessä löytämään vastauksia siihen, millä tavoin suomalaisissa peruskouluissa opetetaan ohjelmointia. Aihepiiriä kannattaa siis tutkia ohjelmoinnillisen ajattelun rakentumisen lähtökohdista.

4. Ohjelmoinnin ja ohjelmoinnillisen ajattelun opettaminen

Tässä luvussa tarkastellaan ohjelmoinnin opetuksen teoriaa ja ilmentymistä suomalaisissa peruskouluissa. Ensimmäisessä alaluvussa käsitellään ohjelmoinnin opetusta Perusopetuksen opetussuunnitelman perusteiden (OPH, 2014) näkökulmasta. Toisessa alaluvussa käsitellään ohjelmoinnin opetuksen nykytilaa ja suuntia suomalaisissa peruskouluissa. Viimeisessä alaluvussa käsitellään ohjelmoinnin opetuksessa ilmeneviä haasteita.

4.1 Ohjelmoinnin opetus opetussuunnitelmassa

Ohjelmoinnin opetus koulumaailmassa sai alkunsa 1960-luvulla Logo-ohjelmointikielen kehityksen myötä (Kalelioglu & Gülbahar, 2014). Ohjelmoinnin opetuksen liittämisestä opetussuunnitelmiin on esitetty maailmalla ehdotuksia ja suosituksia niin ikään 1960-luvulta alkaen esimerkiksi ACM:n *(Association for Computing Machinery)* ja IEEE *(Institute of Electrical and Electronics Engineers)* Computer Societyn toimesta (Impagliazzo, Campbell-Kelly, Davies, Lee & Williams, 1998). Suomessa ohjelmointi sisällytettiin osaksi perusopetuksen opetussuunnitelmaa (OPH, 2014) vuonna 2016. Ennen tätä ohjelmoinnin opetus suomalaisissa peruskouluissa on ollut hyvin koulukohtaista, ja se on saattanut lukeutua esimerkiksi osaksi ATK-opetusta tai sitten sitä ei ole opetettu lainkaan (Saarikoski, 2006).

Suomen perusopetuksen valtakunnallisessa opetussuunnitelmassa ohjelmointi on sisällytetty osaksi laaja-alaisten osaamistavoitteiden tieto- ja viestintäteknologista osaamista (L5), sekä matematiikan ja käsitöiden sisältöjä ja tavoitteita (ks. taulukko 1). Algoritminen ajattelu – tässä tutkielmassa laajemmin ohjelmoinnillinen ajattelu – on sisällytetty osaksi vuosiluokkien 7.–9. matematiikan sisältöjä ja tavoitteita. Ohjelmoinnillisen ajattelun käsitepiiriin tulkittavia ilmauksia kuitenkin löytyy Perusopetuksen opetussuunnitelman perusteista (OPH, 2014) jo aiemmiltakin vuosiluokilta. Esimerkiksi vuosiluokkien 1.–2. matematiikan sisällöissä kohdassa *S1 Ajattelun taidot* todetaan ohjelmoinnin alkeiden alkavan vaiheittaisten toimintaohjeiden laatimisesta, joka voidaan tulkita algoritmisen – tai ohjelmoinnillisen ajattelun – harjoitteluksi.

Ohjelmoinnin opetus aloitetaan perusopetuksessa jo alkuopetuksen vuosiluokilta 1.–2. Ohjelmointia lähestytään matematiikan alkuopetuksen keinoin, eli konkretian, toiminnallisuuden, kokemuksellisuuden sekä aistillisuuden avulla ja sitä harjoitellaan vaiheittaisia toimintaohjeita

antamalla (ks. taulukko 1). Myös pelillisyyttä käytetään oppimisen edistäjänä ohjelmoinnin yhteydessä. Vuosiluokkien 3.–6. aikana ohjelmointia lähestytään innostamisen, kokeilemisen, kokemuksellisuuden sekä kokonaisten ohjelmointiprojektien kautta, ja sitä harjoitellaan graafisissa ohjelmointiympäristöissä sekä robotiikkaa hyödyntäen.

Perusopetuksen opetussuunnitelman perusteissa (OPH, 2014) käsityön sisällöissä ohjelmoinnin yhteydessä puhutaan myös kokeilemisesta ja kehittämisestä. Tämä voidaan lukea yhdeksi ohjelmoinnillisen ajattelun harjoittamisen ilmenemäksi ohjelmoinnin yhteydessä (ks. Brennan & Resnick, 2012). Vuosiluokilla 7.–9. ohjelmointia harjoitellaan osana eri oppiaineiden opintoja, painottuen kuitenkin matematiikan opetukseen. Matematiikan sisällöissä ohjelmoinnin yhteydessä mainitaan sen käytöstä ongelmanratkaisuun, hyvien ohjelmointikäytäntöjen harjoittelu sekä ohjelmointi ja sen soveltaminen itsetehtyihin tai valmiisiin tietokoneohjelmiin. Näillä vuosiluokilla myös kehitetään, harjoitellaan ja sovelletaan ohjelmoinnillista ajattelua. Käsitöissä ohjelmointia sovelletaan sulautettujen järjestelmien suunnitelmiin ja valmiisiin tuotteisiin. (OPH, 2014.) Perusopetuksen opetussuunnitelman perusteissa ohjelmointiin liittyvät ilmaukset on esitetty taulukossa 1.

TAULUKKO 1. Perusopetuksen opetussuunnitelman perusteista löytyvät maininnat ohjelmoinnin opetukseen liittyen. (OPH, 2014)

Vuosiluokka	**Oppiaineet**	**Lähestymiskeinot**	**Sisällöt**	**Laaja-alaiset**
1.–2.	Matematiikka	Konkretia, toiminnallisuus, kokemuksellisuus ja aistillisuus.	(S1) Vaiheittaiset toimintaohjeiden laatiminen, joita testataan (T12) Vaiheittaisten toimintaohjeiden laatiminen, ohjeen mukaan toimiminen.	(L5) Oppilaat saavat ja jakavat keskenään kokemuksia digitaalisen median parissa työskentelystä sekä ikäkaudelle sopivasta ohjelmoinnista. Pelillisyyttä hyödynnetään oppimisen edistäjänä.
3.–6.	Matematiikka	Innostaminen, kokeileminen, kokemuksellisuus, kokonaiset	(S1) Toimintaohjeiden laatiminen graafisessa ohjelmointiympäristössä	(L5) Oppilaat oppivat käyttämään erilaisia laitteita, ohjelmistoja ja palveluita sekä

		ohjelmointiprojektit.	(T14) Suunnitellaan ja toteutetaan ohjelmia graafisessa ohjelmointiympäristössä.	ymmärtämään niiden käyttö- ja toimintalogiikkaa. Ohjelmointia kokeillessaan oppilaat saavat kokemuksia siitä, miten teknologian toiminta riippuu ihmisen tekemistä ratkaisuista.
	Käsityö		(S3) Harjoitellaan ohjelmoimalla aikaan saatuja toimintoja, joista esimerkkinä robotiikka ja automaatio. Kokeilujen pohjalta tuotetta tai teosta kehitetään vielä eteenpäin.	
7.–9.	Matematiikka	Kehitetään, harjoitellaan ja sovelletaan algoritmista ajattelua.	(S1) Ohjelmoidaan ja samalla harjoitellaan hyviä ohjelmointikäytäntöjä. Sovelletaan itse tehtyjä tai valmiita tietokoneohjelmia osana matematiikan opiskelua. (T20) Ohjata oppilasta kehittämään algoritmista ajatteluaan sekä taitojaan soveltaa matematiikkaa ja ohjelmointia ongelmien ratkaisemiseen.	(L5) Ohjelmointia harjoitellaan osana eri oppiaineiden opintoja.
	Käsityö		(S3) Käytetään sulautettuja järjestelmiä käsityöhön eli sovelletaan ohjelmointia suunnitelmiin ja valmistettaviin tuotteisiin.	

Myös muualla maailmassa ohjelmointi on sisällytetty osaksi opetusta ja virallisia opetussuunnitelmia viimeisten vuosien aikana. Ennen Suomea ohjelmointi on otettu mukaan kansalliseen opetussuunnitelmaan esimerkiksi Englannissa, Australiassa sekä Virossa. Joissain maissa ohjelmointi on saavuttanut aseman omana oppiaineenaan, joskin oppiaineen toteutus vaihtelee ohjelmoinnin, tietojenkäsittelytieteen ja ohjelmoinnillisen ajattelun kehityksen välillä. Useissa maissa, kuten Suomessa, ohjelmointi on integroitu osaksi muiden oppiaineiden sisältöjä. (Balanskat & Engelhardt, 2015.) Yhteneväistä eri maiden välillä on kuitenkin ohjelmoinnin tai ohjelmoinnillisen ajattelun viimeaikainen lisääntynyt ilmenevyys kansallisissa opetussuunnitelmissa. Useissa Euroopan maissa ohjelmoinnin opetuksen lisäämisellä osaksi opetussuunnitelmia pyritään lisäämään oppilaiden kykyä loogiseen ajatteluun ja ongelmanratkaisuun ohjelmoinnillisen ajattelun kehityksen kautta. (Bocconi ym., 2016.)

Perusteita ohjelmoinnin integroimiselle peruskoulujen opetussuunnitelmiin on esitetty useita. Pääpaino integroinnin perusteissa tuntuu löytyvän kuitenkin oppilaiden ajattelun taitojen kehityksestä sekä työelämän tarpeiden piiristä, (Vuorikari, Punie, Carretero & Van den Brande, 2016; Balanskat & Engelhardt, 2015) ja erityisesti ohjelmoinnillisen ajattelun kehittyminen on nähty olennaisena tekijänä (Bocconi ym., 2016). Esimerkiksi Euroopan komission (Vuorikari ym., 2016) julkaisema *The Digital Competence Framework for Citizens* sisältää näkemyksiä digitaalisten taitojen – joihin myös ohjelmointi lukeutuu – tärkeydestä ihmisten osallisuuden ja toimijuuden vahvistamiseksi digitalisoituvassa yhteiskunnassa.

Digitaalisista taidoista puhutaankin joskus nimellä *digitaalinen lukutaito*, joka terminä pitää sisällään sekä ohjelmoinnin ja ohjelmoinnillisen ajattelun (Madigan & Martin, 2006). Samaa näkemystä on jaettu myös kansallisella tasolla esimerkiksi Opetus- ja kulttuuriministeriön (2013) toimesta. Digitaalisten taitojen tärkeyttä on perusteltu sekä yksilön ajattelun taitojen ja toimijuuden vahvistamisen näkökulmista että myös tulevaisuuden työelämän tarpeiden kautta (Vuorikari ym., 2016). Ohjelmointiosaamisella ja ohjelmoinnin opetuksella onkin nähty olevan vahvoja yhteyksiä näiden tavoitteiden saavuttamiseen, sillä ohjelmointiosaamisen nähdään lisäävän yksilön mahdollisuuksia hallita ja luoda teknologisia sisältöjä ja näin kukoistaa yksilönä ja yhteiskunnallisena toimijana. Ohjelmoinnin opetusta onkin liitetty eri maiden opetussuunnitelmiin viime vuosien aikana laajalti. (Balanskat & Engelhardt, 2015.)

4.2 Ohjelmoinnin opetuksen nykytila suomalaisissa peruskouluissa

Vuonna 2016 käyttöön otettujen Perusopetuksen opetussuunnitelman perusteiden (OPH, 2014) sisällyttää velvoitteen ohjelmoinnin opettamisesta. Tämä ei ole vielä kuitenkaan tarjonnut riittäviä lähtökohtia yhtenäiselle ohjelmointiopetukselle maanlaajuisesti (OPH, 2021). Osana Opetushallituksen (2021) Oikeus oppia -kehittämisohjelmaa (2020–2022) laaditun Uudet lukutaidot -kehittämisohjelman avulla ohjelmoinnin ja digitaalisten taitojen opetusta pyritään yhdenvertaistamaan maanlaajuisesti. Sen tavoitteena on vahvistaa lasten ja nuorten medialukutaitoja, tieto- ja viestintäteknologisia taitoja sekä ohjelmoinnin osaamista.

Kokonaisuudessaan Uudet lukutaidot- kehittämisohjelma voidaan nähdä laajentavan ja sanallistavan Opetushallituksen (2014) Perusopetuksen opetussuunnitelman perusteiden sisältämiä kohtia ohjelmoinnista ja algoritmisesta ajattelusta. Kehittämisohjelmaa on tuottamassa alueellisten asiantuntijoiden lisäksi kasvatuksen ja opetuksen järjestäjiä ja toteuttajia. Ohjelmointiosaamisen suhteen kuvaukset on jaettu kolmeen pääalueeseen: 1) *ohjelmoinnillinen ajattelu*, 2) *tutkiva työskentely ja tuottaminen* sekä 3) *ohjelmoidut ympäristöt ja niissä toimiminen* (ks. taulukko 2). Näiden kuvausten sanotaan muodostavan kehittyvän osaamisen polun varhaiskasvatuksesta perusopetuksen loppuun saakka. Uudet lukutaidot -kehittämisohjelma sisältää ”*hyvän ohjelmointiosaamisen kuvauksia*”, jotka on jaoteltu vuosiluokka-asteille erikseen. (OPH, 2021.) Tällaisen kehittämisohjelmahankkeen voidaan nähdä esittävän kattavan kuvan siitä, mihin suuntaan tässä tapauksessa ohjelmoinnin opettamista halutaan Opetushallituksen taholta suunnata. Uudet lukutaidot -kehittämisohjelman mukaiset kuvaukset ohjelmointiosaamisesta perusopetuksen luokka-asteilla 1–9 on esitelty mukaillen taulukossa 2.

TAULUKKO 2. Uudet lukutaidot -kehittämisohjelman kuvauksia ohjelmointiosaamisesta (OPH, 2021)

	1.–2.	3.–6.	7.–9.
	Ohjelmoinnillinen ajattelu		
Looginen ajattelu ja tiedon käsittely	Järjestely ja vertailu erilaisten ehtojen perusteella Säännönmukaisuuksien tunnistaminen ja havaintojen tekeminen	Järjestely ja vertailu käsitteitä ja symboleja käyttäen Kokonaisuuksien hahmottaminen	Yleistyksiin sisältyvien tietojen käsittely Erilaisten merkintätapojen käyttö

	Havaintojen ja valintojen ilmaiseminen käsitteiden ja konkreettisten välineiden avulla	Asioiden keskinäisten yhteyksien löytäminen ja kuvaaminen	Erityyppisten tietojen käyttö loogisissa operaatioissa
Ongelmien ratkaiseminen ja mallintaminen	Arjen ilmiöihin liittyvien ongelmien purkaminen osiin Ratkaisuvaihtoehtojen etsiminen Ratkaisutavoista kertominen	Erilaisten tapojen käyttäminen ja itse keksiminen ongelmanratkaisussa Näiden tapojen arviointi jonkin kriteerin perusteella	Ongelmien analysointi ja erilaisten ratkaisujen arviointi erilaisten kriteerien perusteella Ongelmien ja ratkaisujen visualisointi yleistyksillä ja kaavioilla
Ohjelmoinnin käsitteet ja perusrakenteet	Vaiheittaisten toimintaohjeiden laatiminen komentoja ja toistorakenteita hyödyntäen Virhetilanteiden tunnistaminen ja ratkaisutapojen kokeilu	Täsmällisten ja yksityiskohtaisten toimintaohjeiden käyttö toisto- ja valintarakenteita hyödyntäen Virheiden etsiminen ja korjaaminen toimintaohjeista	Algoritmin merkityksen ymmärrys Tarkoitusmukaisesti ohjelmoinnin perusrakenteita hyödyntävän ohjelman suunnittelu
Käytännön taidot	Leikillisten toimintaohjeiden laatiminen toisille oppilaille, laitteelle tai sovelluksessa	Toimintaohjeiden laatiminen laitteettomasti sekä laitteita ohjaten Oman ohjelman laatiminen graafisessa ohjelmointiympäristössä	Ohjelmien ohjelmointi eri ympäristöissä Perusasioiden ymmärtäminen ja tulkintakyky yhdestä tekstipohjaisesta ohjelmointikielestä
	Tutkiva työskentely ja tuottaminen		
Yhteiskehittelyn prosessit	Omien ideoiden esittely Toisten ideoiden huomioiminen Yhteinen ratkaisujen kokeileminen Ohjelmointitehtävissä roolien vuorottelu	Omien ajattelutapojen kuvailu Toisten näkökulmien huomioiminen Työskentely yhteisten tavoitteiden saavuttamiseksi ohjelmointiprojekteissa	Ymmärrys erilaisista ryhmärooleista ja yhteistyön tavoista Vastavuoroinen työskentely ja aktiivinen osallistuminen ohjelmointiprojekteissa
Luova tuottaminen	Mallinnusten rakentelu ohjeiden mukaisesti ja	Omien havaintojen ja mittausten hyödyntäminen	Antureita ja automaatiota hyödyntävän ratkaisun

	luovasti omia ideoitaan toteuttaen Ideoiden esittely ja jako Tarinallisia tai pelillisiä elementtejä sisältävien digitaalisten tuotosten animointi tai yksinkertainen ohjelmointi ohjatusti tai yhteistyössä	omissa tuotoksissa sekä niiden yhdistäminen robotiikkaan ja automaatioon Olemassa olevien ratkaisujen jalostaminen Animaatioiden ja pelien ohjelmoinnillisten piirteiden tunnistaminen Pelien tekeminen erilaisilla animointi- ja ohjelmointialustoilla	suunnitteleminen ja luominen yhteistyössä prosessinomaisesti Pelin, simulaation tai sovelluksen toteuttaminen, jolla yhteys oppiaineeseen tai oikean elämän ongelmaan
Ohjelmointi oppimisen välineenä	Eri oppiaineissa harjoiteltavien sisältöjen käsittely ohjelmointiin liittyvin toimintatavoin ja välinein	Ohjelmointiin liittyvien työtapojen ja -välineiden käyttö luovaan ilmaisuun, omaan tuottamiseen, erilaisten ilmiöiden tutkimiseen ja selittämiseen oppiaineissa ja monialaisissa oppimisprojekteissa	Tuntemus eri oppiaineisiin liittyvistä teknologisista sovelluksista ja niiden toimintaperiaatteiden selittäminen Algoritmisen ajattelun ja ohjelmoinnin hyödyntäminen eri oppiaineissa ja projekteissa
	Ohjelmoidut ympäristöt ja niissä toimiminen		
Ohjelmoitu teknologia elämän eri osa-alueilla	Omasta kokemusmaailmasta löytyvän tietotekniikan tunnistaminen ja nimeäminen Tutustuminen robotiikkaan Laitteiden käyttötarkoitusten ja toimintaperiaatteiden kuvailu	Ohjelmoinnin ja robotiikan havainnointi ympäröivässä yhteiskunnassa Teknologisten sovellusten hyödyntämistapojen ja toimintaperiaatteiden kuvailu ja merkityksen tunnistaminen omassa elämässä	Algoritmien, automaation ja robotiikan toimintalogiikan ja sovellusten tuntemus Pohdinta teknologian mahdollisuuksista, riskeistä ja eettisistä näkökulmista
Ohjelmoidun teknologian vaikutukset arjessa	Pohdinta, ymmärrys ja esimerkkien anto siitä, mil-	Esimerkkien kertominen kohdennetuista digitaalisista sisällöistä sekä kohdentamisen tavoista	Esimerkkien kertominen digitaalisten palveluiden

	laista tietoa itsestä kerätään digitaalisissa ympäristöissä	Pohdinta oman toiminnan ja siitä kerääntyvän tiedon käytöstä digitaalisissa ympäristöissä	personoinnista ja mainonnan kohdentamisesta käyttäjälle. Pohdinta digitaalisten palveluiden keräämän tiedon ja ohjelmoinnin merkityksestä sosiaalisessa ja yhteiskunnallisessa vaikuttamisessa.

Perusopetuksessa tapahtuvaa ohjelmoinnin opetusta on käsitelty viime vuosina myös useissa opinnäytetöissä. Ne ovat kuitenkin pääasiassa painottuneet tutkimaan opettajien valmiuksia (Ihanainen, 2020; Raivonen, 2018), opetuksen kynnyskäsitteitä (Moilanen, 2018) tai käsityksiä ja mielikuvia (Kurkinen, 2018; Makkonen & Pyykönen, 2018) ohjelmoinnin opetuksesta, tämän tutkielman syventäen näkökulmia esimerkiksi ohjelmoinnin opetuksen käytännön keinoista ja päämääristä.

Ohjelmoinnin opettaminen ei kuitenkaan kehity itsekseen, vaan se vaatii opettajilta riittävää osaamista tieto- ja viestintäteknologisten (TVT) taitojen osalta (Mason & Rich, 2019). Opettajien käsityksiä tieto- ja viestintäteknologisista taidoista on tutkittu ja tulosten mukaan Suomessa opettajien käsitykset heidän omista TVT-taidoistaan ovat matalammalla tasolla verrattuna verrokkiopettajiin Kiinassa, Singaporessa, Taiwanissa ja Etelä-Koreassa. Opettajien käsitykset myös TVT-taitojen hyödyllisyydestä ja koulun tarjoamasta tuesta siihen ovat myönteisemmät muualla edellä mainituista maista kuin Suomessa. Tutkimuksen mukaan niin ohjelmoinnin kuin TVT-taitojenkin osaamisen suhteen on myös sukupuolten välillä eroja. Yleisesti ottaen miesopettajat arvioivat oman osaamisensa korkeammaksi, kuin naisopettajat. (Wu, Looi, Multisilta, How, Choi, Hsu & Tuomi, 2019.)

Myös Digiajan peruskoulu II -hankkeen (2020) mukaan tulos on samansuuntainen sen suhteen, että miehet kokevat olevansa TVT:n asiantuntijoita, kuin suurempi osa naisista kokee olevansa pikemminkin perustasolla. Digiajan peruskoulu II -hankkeen (2020) tavoitteena oli luoda perustaa opetusalan tutkimustiedon hyödyntämiselle valtionhallinnon päätöksenteossa ja selvittää

miten digitalisaatio on edennyt koulun kehittämisen toimintojen eri osa-alueilla. Saman hankkeen (2020) mukaan ohjelmoinnin opettaminen ei vielä ole sisällöltään sillä tasolla, mitä Opetushallituksen (2014) kirjaamat Perusopetuksen opetussuunnitelman perusteet sille ovat asettaneet. Hankkeen tuloksista selviää, että ohjelmoinnin opetus ei ole vielä saanut vakiintuneita muotoja kouluissa, vaan ohjelmoinnillista ajattelua harjoitetaan osana kaikkea opetusta – kuitenkin suunnitelmallisemmin alakoulussa verrattuna yläkouluun. Perusopetuksen opetussuunnitelman perusteissa (2014) useasti mainittu tavoite ”*ohjelmoidaan graafisessa ohjelmointiympäristössä*” ei hankkeen tulosten mukaan toteudu, sillä vain puolet tutkimukseen osallistuneista viidennen vuosiluokan oppilaista ovat käyttäneet graafista ohjelmointiympäristöä oppitunneilla. (Tanhua-Piiroinen, Kaarakainen, Kaarakainen, Viteli, Syvänen & Kivinen 2020.)

Hankkeen tulosten mukaan opettajista kaksi kolmasosaa jäi kokonaan ilman pisteitä heille annetuissa alkeisohjelmoinnin tehtävissä, jotka edellyttivät lähinnä päättelykykyä, selkokielisten komentojen antamista ja yksinkertaisten laskutoimitusten ratkaisemista. Samasta tehtävästä peräti 89 prosenttia yhdeksännen vuosiluokan oppilaista eivät saaneet yhtäkään pistettä. Tuloksiin vaikuttanee se, että tutkimuksen kymmenestä haastatellusta koulusta peräti neljässä ei ohjelmoinnin opetus toteutunut vielä millään tavalla johtuen opettajien puutteellisesta osaamisesta sen suhteen. Hankkeen aineistohaastatteluista saatujen tulosten perusteella opettajien nähdään kärsivän niin sanotusta ”digiähkystä”, jossa uusien ohjelmien ja sovellusten käyttöön ottaminen koetaan lamaannuttavana. (Tanhua-Piiroinen ym., 2020.)

Suomalaisen koulun ohjelmoinnin opettamisesta puhuttaessa esiin nousee usein myös keskustelut STEAM-koulutuksesta. STEAM koostuu englanninkielisistä sanoista *science (suom. tiede), technology (suom. teknologia), engineering (suom. insinööritaidot), arts (suom. taiteet)* ja *mathematics (suom. matematiikka)*. Ohjelmointitaitoa voidaankin pitää perustavanlaatuisena taitona erityisesti STEAM-agendaa tukevissa koulutuksissa ja koulutusmenetelmissä (Wu ym., 2019). STEAM-agendan nähdään kehittämään oppilaiden tietoja ja taitoja, jotka ovat merkityksellisiä ja hyödyllisiä tulevaisuuden työpaikkojen kannalta. STEAM-pohjaisessa opetuksessa korostuu ongelmanratkaisukeskeisyys, jonka avulla pyritään ratkaisemaan maailman kiireellisiä kysymyksiä liittyen innovaatioon. luovuuteen, kriittiseen ajatteluun, tehokkaaseen viestintään, yhteistyöhön ja uuteen tietoon. STEAM-pohjaista opetusta voidaan kuvata monitieteelliseksi opetukseksi, joka nähdään hyvänä lähestymistapana tulevaisuuden taitojen omaksumiselle. (Quigley & Herro, 2016.)

4.3 Ohjelmoinnin opetukseen liittyvät haasteet

Ohjelmointiin peruskoulun kontekstissa liittyy myös erinäisiä haasteita, joita avataan tarkemmin tässä luvussa. Haasteita esiintyy niin opettajien valmiuksien kuin materiaalien ja työ- ja elinkeinoelämän vaikutusten suhteen.

Ohjelmoinnin opettaminen vaatii opettajan, joka ymmärtää ohjelmoimisesta ja sen hyödyllisyydestä. Opettajat tarvitsevat tukea valmiuksien kohottamiseen ja eteen tulevien haasteitten käsittelyyn (Wu ym., 2019). Masonin ja Richin (2019) kirjallisuuskatsauksessa tarkasteltiin alakoulun opettajien valmiuksia ohjelmointiin liittyvien asioiden opettamisessa. Heidän tutkimuksensa aineiston mukaan huonommat teknologisen osaamisen taidot omaavat opettajat käyttivät teknologiaa harvemmin opetuksessaan, kuin teknologian käytön suhteen itsevarmat opettajat. Kirjallisuuskatsaus osoitti myös, että mikäli opettajalla olisi valmiudet ohjelmoinnin opettamiseen, mutta kokemus ohjelmoinnin opetuksesta turhana, ei hän tällöin todennäköisesti opeta sitä. Sama lopputulema on myös tapauksessa, jossa opettaja kokee ohjelmoinnin opettamisen hyödyllisenä, mutta ei omaa riittäviä valmiuksia sen opettamiseen. Kirjallisuuskatsauksen mukaan koulutuksen puutteen vuoksi opettajat kohtaavat tunnesidonnaisia ja tiedonpuutteeseen liittyviä esteitä.

Myös Iso-Britanniassa toteutetun tutkimuksen mukaan opettajien suurimpina haasteina opetuksessa mainittiin oman tietämyksen puute. Lisäksi haasteita koettiin eriyttämisessä, lähestymistapojen ja arviointimenetelmien valitsemisessa sekä opetukseen liittyvässä tuessa. Myös resurssipulat liittyen laitteistoon ja ohjelmistoihin mainittiin haastavana. Opiskelijoiden suhteen haasteita olivat aiheen ymmärtämisen vaikeuksien lisäksi ongelmanratkaisutaitojen ja matemaattisten taitojen puuttuminen. (Sentace & Czizmadia, 2016.)

Mertalan, Palsan ja Dufvan (2020) artikkelin mukaan ohjelmoinnin opettamisesta puhuttaessa on tärkeää pohtia, kenen ehdoilla ja millaisia päämääriä varten sitä opetetaan. Artikkelissa pohditaan ja pyritään laajentamaan nykyisen ohjelmoinnin opetuksen funktionaalista näkökulmaa ottamalla huomioon myös yhteiskunnalliset ja muut ulottuvuudet. Toisin sanoen artikkeli sukeltaa ohjelmoinnin opettamisen lähtökohtiin melko syvällisellä tasolla. Kirjoittajien mukaan laajentamalla ohjelmoinnin opetuksen määritelmää päästään pois niin sanotusta koulu hoitaa -diskurssista, jossa kaikkien sekä yhteiskunnallisten että yksityisten epäkohtien korjaamisen ajatellaan olevan pelkästään koulun vastuulla. Monilukutaito koodin purkajana -artikkeli (Mertala,

Palsa & Dufva, 2020) nostaa esiin ohjelmoinnin opettamiseen liittyviä eri lähestymistapoja: funktionaalinen ja emansipatorinen ohjelmointikäsitys. Näistä ensimmäisellä tarkoitetaan käsitystä, jossa koodi nähdään vain tietokoneelle annettuina peräkkäisinä ohjeina, jotka tietokone käsittelee. Jälkimmäisessä, emansipatorisessa käsityksessä ohjelmoinnin opettamisessa otetaan huomioon ympäristön valtavaikutussuhteet ja niiden esiintyminen koodissa. Emansipatorisessa ohjelmointikäsityksessä tuleekin pohtia, lisääkö vai rajoittaako koodi vapautta sekä yksilön että yhteiskunnan tasolla. (Dufva & Dufva, 2016.) Monilukutaito koodin purkajana -artikkelissa kuitenkin painotetaan, että molempia ohjelmointikäsityksiä tarvitaan, sillä ne nähdään välttämättöminä toisiaan täydentävinä tulokulmina, joita ei tule käsitellä erikseen. (Mertala ym. 2020.)

Ohjelmoinnin opetusta ja sen merkitystä on tarkasteltu kriittisesti myös 2015 julkaistussa artikkelissa, jossa esitetään ohjelmoinnillisen ajattelun kehittymisen tukemisen pahimmassa tapauksessa vaarantavan tulevan sukupolven syrjäyttämällä muita avaintaitoja, kuten luku- ja kirjoitustaitoja. Artikkelin mukaan tarvitaan enemmän teoreettista ja käytännön tutkimusta ohjelmoinnillisen ajattelun kehittymisestä, jotta se voidaan perustellusti sisällyttää opetussuunnitelmiin. Ohjelmoinnin myönnetään rikastuttavan koulutusta, mutta sen opetukselle tulee olla tarkat perustellut, miksi sille tulisi antaa painoarvoa tulevaisuutta ajatellen. (Bresnihan, Millwood, Oldham, Strong & Wilson 2015.)

Tutkimusten mukaan suunta koulun kehittymiselle löytyy usein koulun ja kasvatustieteen ulkopuolelta – markkinoilta ja työelinkeinoelämästä. Näin ollen näiden vaikutukset koululaitosta kohtaan voidaan nähdä haasteina, jotka liittyvät esimerkiksi materiaalien tuottamiseen tai saamiseen koulun opetukseen. Koulujen tiedetään hyödyntävän paljon yritysten materiaaleja, jotka osaltaan vaikuttavat esimerkiksi koulujen opetussuunnitelmiin ja niiden sisältöihin. (Bresnihan ym., 2015.) Mertalan, Palsan ja Dufvan (2020) artikkelissa pohditaankin esimerkiksi eri yritysten osallisuutta ohjelmoinnin koulutustarjonnan suhteen – muokkaavatko yksityistä täydennyskoulutusta tarjoavan yrityksen ideologiat ja intressit liikaa koulun käytäntöjä opettaa ohjelmointia.

5. Tutkimuksen toteutus

Tässä luvussa esitellään tutkielman toteutus. Ensimmäisessä alaluvussa käsitellään tutkimuskohdetta ja lähestymistapaa. Tutkimuksen vastaajajoukkoa käsitellään ja kuvaillaan toisessa alaluvussa. Kolmannessa alaluvussa kerrotaan tutkimuksen aineistonkeruusta ja viimeisessä eli neljännessä alaluvussa käsitellään aineiston analyysimenetelmiä ja sen eri vaiheita tässä tutkielmassa.

5.1 Laadullinen tapaustutkimus

Tutkielmassa tutkittiin ohjelmoinnin opettamista peruskoulukontekstissa. Tutkielman tavoitteena oli selvittää ja määritellä, millaista ohjelmoinnin opetus on suomalaisissa peruskouluissa. Tutkimuskysymyksiin pyrittiin vastaamaa analysoimalla aineistoa, jota kerättiin sähköisen kyselylomakkeen avulla. Aineistoa analysoitiin ja käsiteltiin teoriaohjaavan sisällönanalyysin keinoin. Tutkielmaa voidaan kuvailla monimenetelmälliseksi tapaustutkimukseksi painottuen kvalitatiiviseen tutkimukseen, jonka aineistoa käsiteltiin sekä laadullisia että määrällisiä analyysimenetelmiä hyödyntäen.

Tutkielma on pääpiirteissään laadullinen eli kvalitatiivinen tutkimus. Tutkielman analyysissä hyödynnettiin kuitenkin myös määrällisiä, eli kvantitatiivisia analyysin keinoja. Kvalitatiiviselle tutkimukselle on tyypillistä ihmiskeskeinen asetelma. Laadullisella tutkimuksella pyritään tutkimaan ihmistä, hänen näkemyksiään ja kokemuksiaan sekä hänen elämäänsä hyödyntäen erilaisia traditioita, lähestymistapoja sekä aineistonkeruu- ja analyysimenetelmiä. Laadullinen tutkimus pyritään toteuttamaan ilman hypoteeseja ja asettamatta tutkittavia erilaisiin kokeellisiin asetelmiin. Tällä tavoin tutkimuksessa pyritään naturalismiin. (Puusa & Juuti, 2020.) Tutkimuksen aineisto ja analyysi eivät ilmene numeraalisina, vaan tekstinä. Tämä ei kuitenkaan poissulje niiden määrällistä analysointia. Kvalitatiivisella tutkimuksella tarkoitetaan laadullista tutkimusta, jonka tavoitteena on ymmärtää ja kuvata tiettyä ilmiötä (Kiviniemi, 2007).

Tutkielman viitekehys ja lähestymistapa huomioiden tutkielma on tapaustutkimus, jossa on piirteitä monimenetelmällisyydestä. Tapaustutkimuksessa tutkimuksen kohde on usein jokin tapahtumakulku tai ilmiö. Sille on ominaista tarkastella pientä joukkoa tapauksia tai vain yhtä tapausta. Tapaustutkimuksella pyritään perusteellisuuteen ja siten antamaan tarkkapiirteinen

kuvaus tutkittavasta ilmiöstä. Sitä voidaan luonnehtia myös hyvin kokonaisvaltaiseksi tutkimukseksi, sillä siihen sisältyy usein erilaisten aineistojen ja menetelmien käyttämistä yhdistäen niitä aikeisempiin tutkimuksiin. Yksi keskeisistä päämääristä tapaustutkimukselle on lisätä ymmärrystä tutkittavasta tapauksesta vastaamalla ilmiöön liittyviin kysymyksiin: *miten* ja *miksi*. (Laine ym., 2007.)

Tapaustutkimuksessa usein hyödynnetään sekä laadullista että määrällistä aineistoa, joka johtaa myös erilaisiin analyysitapoihin. Tästä syystä tapaustutkimuksen voidaan ajatella olevan läheisessä suhteessa monimenetelmälliseen tutkimukseen (*eng. mixed methods research*). Tapaustutkimus edustaa laadullista osaa tässä tutkielmassa, mutta tutkielmassa hyödynnetään myös monimenetelmällistä tutkimusotetta. Taitavasti toteutettuna monimenetelmällisessä tutkimusotteessa laadulliset ja määrälliset menetelmät täydentävät toisiaan paikkaamalla mahdollisia ongelmakohtia, joita syntyisi käytettäessä vain toista menetelmää. (Eriksson & Koistinen, 2014; Tuomi & Sarajärvi, 2018). Monimenetelmällisyydellä (*eng. mixed methods*) pyritään luomaan parempaa ymmärrystä tutkimusongelmiin (Tuomi & Sarajärvi, 2018).Tässä tutkielmassa monimenetelmällisyyden kvantitatiivinen puoli ilmenee lähinnä tulosten analyysissä frekvenssianalyysin keinoin.

Puusan ja Juutin (2020) mukaan laadullisen tutkimuksen alueelta on tunnistettavissa vain vähän standardoituja tapoja analysoida aineistoa. Eri analyysitavat eivät sinällään ole toisiaan parempia, ja lähestymistavan valintaan tulisikin vaikuttaa sen tarkoituksenmukaisuus. Tutkija itsessään toimii tutkimusinstrumenttina. Tutkimuksessa on hyödytöntä pyrkiä täysin arvovapaaseen laadulliseen tutkimukseen, vaan ennemminkin tulisi tunnistaa oma esiymmärrys ja erottaa se aineistosta. Aineistoa käsitellään jatkuvasti tutkimuksen alusta loppuun ja tutkija pyrkii siitä saamiensa vihjeiden avulla aineiston ryhmittelemiseen eri teemoihin, luokkiin ja kategorioihin. (Puusa & Juuti, 2020.)

Ohjelmoinnin opetusta ja sen merkitystä on tarkasteltu kriittisesti myös 2015 julkaistussa artikkelissa, jossa esitetään ohjelmoinnillisen ajattelun kehittymisen tukemisen pahimmassa tapauksessa vaarantavan tulevan sukupolven syrjäyttämällä muita avaintaitoja, kuten luku- ja kirjoitustaitoja. Artikkelin mukaan tarvitaan enemmän teoreettista ja käytännön tutkimusta ohjelmoinnillisen ajattelun kehittymisestä, jotta se voidaan perustellusti sisällyttää opetussuunnitel-

miin. Ohjelmoinnin myönnetään rikastuttavan koulutusta, mutta sen opetukselle tulee olla tarkat perustellut, miksi sille tulisi antaa painoarvoa tulevaisuutta ajatellen. (Bresnihan ym., 2015.) Aineiston purkamisen jälkeen edessä häämöttää tutkimuksen tekemisen ”jyrkin mäki” eli analyysivaihe. Miten aineistosta edetään kelvolliseen analyysiin ja sitä kautta tieteelliseen eli tutkimukseen kelpaavaan? Analyysivaiheen jälkeen kuitenkin seuraa pitkä ja loiva alamäki, jolloin aineistoa hiotaan, tarkistetaan ja editoidaan osana työn valmistumista. Eskolan (2018) mukaan aineistosta ei nouse esiin yhtäkään tulosta ilman tutkijan aktiivista työstämistä aineiston analyysin ja tulkinnan kanssa. Laadullista aineistoa tutkiva joutuu tekemään paljon erilaisia valintoja myös analyysin suhteen tutkimuksen edetessä – jotkin isompia ja jotkin pienempiä. Osa valinnoista tulee tehdä ennen aineiston keräämistä, kuten se, miten aineistoa analysoidaan. Näiden valintojen tekemisen keskiössä olennaista on se, että tutkija perustelee ja pohtii niitä myös raportissaan. Tärkeintä on tutkijan ymmärrys siitä mitä tekee. Laadullisen tutkimuksen ongelma voi kummuta teorian ja empirian välisestä kuilusta, sillä laadullisesta aineistosta harvemmin on nostettavissa sellaisia tuloksia, jotka voidaan esitellä selkeinä tuloksina ilman viittauksia teoriaan ja aiempiin tutkimuksiin. Tutkimusten tulee aina käydä keskustelua aiempien kirjoittajien ja tutkijoiden kanssa, eikä vain perustua uusien tulosten esittämiseen. (Eskola 2018.)

Tämän pro gradu -tutkielman laadullisen analyysin menetelmäksi valittiin teoriaohjaava sisällönanalyysi, joka koettiin toimivimmaksi analyysimenetelmäksi tässä tutkielmassa. Tutkielman analyysin toteutus teoriaohjaavana sisällönanalyysinä auttoi löytämään vastukset asetettuihin tutkimuskysymyksiin. Analyysiä ja sen eri vaiheita kuvataan tarkemmin aineiston analyysin luvussa. Sisällönanalyysille ei ole olemassa olemissa yleismaallisia tai aina päteviä sääntöjä, vaan analyysissä on tutkijan käytettävää omaa kekseliäisyyttään ja tuotettava itse oman analyysinsä viisaus. Ongelmaksi tästä muodostuu aineiston analyysin kannalta siis tutkijan aseman vaikeus. Tutkijan on oltava aineistolle ja siitä löydettävissä oleville tuloksille herkkä, oivaltava ja terävä sekä osittain myös onnekas. (Tuomi & Sarajärvi 2018.)

5.2 Tutkimukseen osallistujat

Tutkimuksessa käytettyyn tutkimuskyselyyn vastasi 43 vastaajaa, joista 42 toimi opettajana ja yksi pedagogisena asiantuntijana. Osa vastaajista kertoi tekevänsä myös muita, kuin opettajan töitä. Vastaajista neljä toimi luokanopettajana 1.–2. luokka-asteella, kaksitoista 3.–6. luokka-

asteella ja 24 aineenopettajana. Kyselyyn vastanneet aineenopettajat kertoivat opettavansa matematiikkaa (16), fysiikkaa (9), käsityötä (8), tieto- ja viestintätekniikkaa (7) sekä suomen kieltä ja kirjallisuutta (1). Yksi vastaajista toimi erityisluokanopettajana, yksin ammattikoulun aineenopettajana, yksi digitaalisen liiketoiminnan johtajana ja yksi pedagogisena asiantuntijana. Kaikki vastaajista olivat työskennelleet opetusalalla pidempään kuin vuoden, ja nykyisessä tehtävässään suurin osa vastasi toimineensa myös pitempään kuin vuoden. Yksi kyselyyn vastanneista ilmoitti kysyttäessä *"millaisia työtapoja olet käyttänyt opettaessasi ohjelmointia?"* ettei ole opettanut ohjelmointia. Hänen vastauksensa poistettiin lopulta aineistosta, ja otannan kooksi jäi lopulta 42 vastaajaa.

Vastaajien iän keskiarvo oli 40,7 (Md=40), nuorimman vastaajan ollessa iältään 26 ja vanhimman 64 (ks. kuvio 4). Vastaajia kyselyyn saatiin kahdestatoista maakunnasta. Vastaajat kokivat oman kykynsä opettaa ohjelmointia peruskoulutasolla asteikolla 0–10 olevan keskimäärin 7,3 (Md=8). Minimiarvoksi kokemukselle omasta osaamisesta opettaa ohjelmointia annettiin 2 ja maksimiarvoksi 10. Vastaajat kokivat oman koulunsa muiden opettajien kyvyn opettaa ohjelmointia peruskoulutasolla asteikolla 0–10 olevan keskimäärin 4,9 (Md=5), minimiarvon ollessa 0 ja maksimiarvon ollessa 8. (ks. taulukko 3.) Vastaajista *(n=43)* 32:lla oli taustallaan jotain ohjelmoinnin opetukseen hyödyksi katsottavia erityisopintoja. Hyödyksi katsottavia opintoja vastaajilla oli korkeakoulujen opintokokonaisuuksista, erilaisista täydennyskoulutuksista ja lyhyistä ohjelmointikursseista.

TAULUKKO 3. Vastaajien demografinen kuvaus

		1.–2.	**3.–6.**	**7.–9.**	**Muu**	**Yht.**
Ikä						
	alle 30		1	2	1	4
	31–40	4	5	10		19
	41–50		3	7	1	11
	yli 50		3	5		8
Yht.		4	12	24	2	42
Hyödyksi katsottavat erityisopinnot		2	8	20	2	32

Kokemus omasta kyvystä opettaa ohjelmointi (ka)	8,3	7,5	7,4	10,0	7,3
Kokemus oman koulun muiden opettajien kyvystä opettaa ohjelmointia (ka)	5,8	4,2	5,1	8,0	4,9

Kyselyyn vastaajien voidaan ajatella olevan todennäköisesti keskimääräistä opettajaa kiinnostuneempia ohjelmoinnin opetuksesta sen vuoksi, että vastaajat ovat tulleet tarkasti valikoitujen jakokanavien kautta. Vastaajat muodostavat melko edustavan otoksen opettajista, jotka ovat varmasti opettaneet ohjelmointia käytännössä.

5.3 Aineiston keruu kyselylomakkeella

Tutkimuksen aineisto kerättiin maalis-huhtikuussa 2021, ja sen kohderyhmään kuuluivat työssä olevat perusopetuksen opettajat. Aineiston keruuseen käytettiin sähköistä kyselylomaketta, Webropolia, sen saavuttavuuden ja taloudellisuuden vuoksi (Hirsjärvi, Remes & Sajavaara, 2010). Kyselyyn oli mahdollista vastata anonyymisti, mutta sen lopussa tarjottiin mahdollisuus osallistua mahdolliseen haastatteluna toteutettavaan jatkotutkimukseen jättämällä yhteystiedoiksi nimi ja sähköpostiosoite. Jatkotutkimusta ei lopulta järjestetty saadun riittävän otannan vuoksi *(n = 42)*. Kyselyn sähköistä linkkiä jaettiin neljään Facebookin ryhmään. Ryhmät ja niiden jäsenmäärät olivat 26. maaliskuuta 2021 tarkasteltuna *Koodausta kouluun (798)*, *Teknisen työn opetus (1291*), *STEAM Oulu (144)* sekä *Tieto- ja viestintätekniikka opetuksessa/ICT in Education (20795)*. Lisäksi kyselyn sähköistä linkkiä jaettiin Innokas-verkoston Pohjois-Pohjanmaan aluekoordinaattorin toimesta *Innokas-verkoston kouluttajille (49), Teknoluokka-verkostolle (33)* sekä *Steam in Oulu -verkostolle (74)*. Kyselyn saavuttavuus oli siis yli 23000. Kyselylomakkeen linkki jaettiin ryhmiin ja verkostoihin kaksi kertaa. Ensimmäisen jaon jälkeen vastaajia saatiin 21, ja toisen jaon jälkeen vastaajia oli lopulta 43. Kyselyn jaon yhteyteen liitettiin lyhyt esittelyteksti kyselyn tarkoituksesta sekä aihepiiriin liittyvä meemi. Kyselyn alkuun liitettiin myös kyselyn tarkoitusta kuvaava teksti. Kyselytutkimuksen jakoon valikoidut

sosiaalisen media ryhmät ja muut verkostot valikoitiin ehkäisemään kyselytutkimuksissa esiintyvää ongelmaa, jossa vastaajat eivät ole perehtyneet aihepiiriin (Hirsjärvi ym., 2010). Edellä mainittujen ryhmien ja verkostojen jäsenten voitiin olettaa kuuluvan tutkimuksen kohderyhmään ja olevan perehtyneitä ja kiinnostuneita ohjelmoinnin opetuksen aihepiiristä.

Kyselytutkimus mahdollistaa vastaajan mahdollisuuden pohtia ja tarkistaa vastauksiaan, joka katsotaan eduksi kyselyn avointen kysymysten vaatiessa vastaajalta ajatustyötä ja melko laajaa vastausta. Kyselytutkimusten haittapuolia ovat muun muassa vastaajien mahdollisen perehtymättömyyden aihepiiriin, vastaajien sitoutuneisuuden puutteen kyselyyn vastaamiseen, väärinymmärrysten syntymisen, avointen kysymysten mahdolliset puutteelliset vastaukset sekä mahdollisen vastausten puutteen. (Hirsjärvi ym., 2010) Aihepiiriin perehtymättömyyden ongelmakohtaan tutkimuksessa pyritään vaikuttamaan jakamalla kyselylinkki valikoituihin sosiaalisen median- sekä muihin verkostoihin, joiden jäsenten voidaan olettaa kuuluvan tutkimuksemme kohderyhmään. Vastaajien sitoutumiseen kyselyyn vastaamiseen huolellisesti pyritään vaikuttamaan kyselyn jaon yhteyteen liitettävällä motivointikirjeellä. Väärinymmärrysten syntyyn ja avointen kysymysten suppeisiin vastauksiin pyritään vaikuttamaan lisäämällä avointen kysymysten alapuolelle kysymyksiä selkeyttävä, sekä ohjeita vastaamiseen kuvaava, aputeksti. Kyselylomaketta pilotoitiin kahteen kertaan tutkimuskurssin aikana. Pilottien aikana kysymykset ja niiden tukitekstit kokivat muutoksia.

Kyselylomake koostuu kahdesta osiosta. Kyselyn ensimmäisellä osiolla kartoitetaan vastaajien taustoja, ja toinen osio koostuu viidestä avoimesta kysymyksestä, joilla pyritään löytämään vastaukset tutkimuskysymyksiin. Vastaajien taustoista kartoitettiin vastaajien työ- ja koulutushistoria, ikä ja toimialue sekä tuntemus omasta kompetenssistaan ohjelmoinnin opettajana. Kyselyn viisi avointa kysymystä olivat kaikki teorian pohjalta määriteltyjä, ja niiden avulla pyrittiin tarkastelemaan ilmiötä ja löytämään vastauksia asetettuihin tutkimuskysymyksiin. Koska lomakehaastatteluissa ei voida kysyä kysymyksiä sen perusteella, mitä olisi mukavaa tai hyödyllistä tietää, olivat kysymykset tarkoituksenmukaisia ja ongelmanasettelun kannalta merkityksellisiä eli perusteltuja (Tuomi & Sarajärvi 2018).

Kyselylomakkeen viisi avointa kysymystä, niiden tukena olleet tekstit ja mihin tutkimuskysymykseen sen avulla pyritään löytämään vastausta, on esitelty taulukossa 4. Kyselylomake sellaisenaan löytyy tutkielman liitteistä.

TAULUKKO 4. Kyselytutkimuksen avoimet kysymykset ja tutkimuskysymykset

Kyselylomakkeen avoin kysymys	Vastattava tutkimuskysymys
Mitä ohjelmointiin liittyviä käsitteitä pidät tärkeimpinä ohjelmoinnin opetuksessa?	1. Mitkä ohjelmointiin liittyvät käsitteet korostuvat ohjelmoinnin opetuksessa?
Millaisten tehtävien avulla olet opettanut ohjelmointia?	2. Millaisten tehtävien ja työtapojen avulla ohjelmointia opetetaan?
Millaisia työtapoja olet käyttänyt opettaessasi ohjelmointia?	2. Millaisten tehtävien ja työtapojen avulla ohjelmointia opetetaan?
Mitkä ovat mielestäsi keskeiset tavoitteet, joiden vuoksi ohjelmoinnin opettaminen peruskouluissa on tärkeää?	3. Millaisia näkemyksiä opettajilla on ohjelmoinnin opetuksen keskeisistä tavoitteista ja vaikutuksista oppilaiden tulevaisuudelle?
Mitä vaikutuksia ohjelmointiosaamisella on oppilaiden tulevaisuudelle?	3. Millaisia näkemyksiä opettajilla on ohjelmoinnin opetuksen keskeisistä tavoitteista ja vaikutuksista oppilaiden tulevaisuudelle?

5.4 Aineiston analyysi teoriaohjaavan sisällönanalyysin keinoin

Tutkielma on pääasiassa laadullinen, joten sitä tulee analysoida laadullisen aineiston analyysin keinoin. Laadulliselle aineistolle ei kuitenkaan ole yksiselitteistä analyysitapaa, vaan tutkija joutuu itse tekemään päätöksiä – mitkä asiat aineistosta hän ottaa mukaan tutkimukseen. Laadullisen aineiston analyysi voidaan kuitenkin joidenkin mallien mukaan jakaa pääpiirteittäin eri vaiheisiin, jotka sisältävät eri työvaiheita. Eskolan (2018) esimerkin mukaan ensimmäinen vaihe on järjestää aineisto teemoittain. Tämän jälkeen siirrytään tekemään varsinainen analyysi, jossa tutkija nostaa huomioita näistä teemoista erikseen. Huomioiden jälkeen tutkija poimii aineistosta tutkimukselleen tärkeimmät ja mielenkiintoisimmat kohdat ja jäsentelee niitä omin sanoin. Seuraavassa vaiheessa näihin kohtiin liitetään mukaan teoriaa ja aiempia tutkimuksia. Laadullisen aineiston analyysin loppuvaihe keskittyykin pitkälti tekstin hiomiseen ja sen editoimiseen lopulliseen muotoonsa. Eskola (2018) korostaa, että työtä kannattaa hioa ensin karkeimmasta kohdasta ja sitten edetä pienempiin ongelmakohtiin. Jossain vaiheessa tutkijan kuitenkin tulee lopettaa aineistonsa pyörittely ja laittaa työlleen piste.

Myös Tuomi ja Sarajärvi (2018) esittelevät oman, tutkija Timo Laineen esittämästä laadullisen tutkimuksen analyysin etenemisrungosta mukaillun versionsa.

1. Päätös aineiston tärkeimmästä sisällöstä.
2. Aineistosta niiden asioiden suodattaminen ja erilleen ottaminen, jotka kuuluvat tärkeimmän sisällön piiriin, samalla kaiken muun pois jättäminen.
3. Jäljelle jääneen aineiston luokittelu, teemoittelu ja tyypittely.
4. Yhteenveto.

Tämän monimenetelmällisen tapaustutkimuksen analyysissä hyödynnettiin pääasiassa teoriaohjaavan sisällönanalyysin keinoja. Aineistoa analysoitiin kuitenkin myös jonkin verran määrällisen analyysin keinoin, erityisesti *ohjelmoinnin opetuksen käsitteitä* analysoidessa. Sähköisenä kyselynä toteutettu aineistonkeruu oli pakottanut tutkijat asettamaan tutkittavat kohteet jo valmiiksi yläkategorioihin ja teemoihin, jotka olivat ohjelmoinnin opetukseen liittyvät *käsitteet*, *käytänteet* ja *päämäärät*. Päätös aineiston tärkeimmästä sisällöstä oli siis jo osittain tehty etukäteen. Aineiston analyysistä kuitenkin jätettiin pois kaikki sinne kuulumaton, jotka nähtiin olevan tutkimuskysymysten kannalta merkityksettömänä sisältönä.

Teoriaohjaavaan sisällönanalyysiin kuuluvia työvaiheita itse aineistolle ovat: 1) redusointi, 2) klusterointi ja 3) abstrahointi (Tuomi & Sarajärvi 2018). Näitä työvaiheita hyödyntäen tämän tutkimuksen aineistoa käsiteltiin pelkistäen, ryhmitellen ja teoreettisia käsitteitä luoden siten, että jäljelle jäävästä materiasta oli mahdollista tehdä loogisia päätelmiä jonkin sopivan päättelytavan mukaisesti. Tämän tutkimuksen teoriaohjaavalle sisällönanalyysille tuntui sopivimmalta valita induktiivinen päättely. Tavoitteena oli siis muodostaa rajatusta havaintojoukosta ja sen vastauksista yleistys, jota voidaan sitten lopulta verrata teoriaan.

Aineistoon tutustumisen jälkeen aloitettiin analysoimaan aineistosta kyselytutkimuksen kysymyksiä yksi kerrallaan pyrkien löytämään vastauksia tutkimuskysymykseen, joka oli kuhunkin kysymykseen liitetty. Ensimmäisenä analysoitiin ohjelmoinnin opetuksen käsitteet, sen jälkeen ohjelmoinnin opetuksen tehtävät ja työtavat ja viimeisenä ohjelmoinnin opetuksen päämäärät ja vaikutukset. Analyysissä käytettiin hyödyksi kvalitatiivisen tutkimuksen työstöohjelmisto *Nvivoa*, joka koettiin hyvin hyödyllisenä työkaluna laadullisen aineiston analysointiin.

Kysymyskohtainen analysointi aloitettiin lukemalla kysymyksellä saatu aineisto läpikotaisin, ja tutustumalla siellä mainittuihin sivustoihin, sovelluksiin, laitteisiin ja muihin selvennystä vaativiin asioihin. Aineistossa oli siis havaittavissa tutkijoille tuntemattomia sovelluksia ja sivustoja, joihin tutkijoiden tuli tutustua ennen analyysin jatkamista. Tämän vaiheen jälkeen aineistosta voitiin siis muodostaa laaja yleiskuva. Tässä vaiheessa vastaajat myös numeroitiin, jotta vastaajien mainintoja ei sekoiteta analyysin eri vaiheissa. Seuraavaksi aineisto lähdettiin redusoimaan eli pelkistämään. Tämän avulla aineistosta löydettiin siellä toistuvia sisältöjä, jotka nähtiin kuvaavan vastauksia, joilla voidaan vastata asetettuihin tutkimuskysymyksiin. Tässä vaiheessa myös yhdisteltiin selkeästi samaa tarkoittavat sisällöt. Esimerkiksi käsitteen *ehto* alle luettiin myös käsitteet *ehtolause, ehtorakenne* ja *if-lause.* Pelkistämisen jälkeen toistuvat sisällöt klusteroitiin eri ryhmiteltiin kuvaaviin alaluokkiin, jotka muodostuivat osittain teorian ja osittain aineiston sisällön pohjalta.

Esimerkki teorian pohjalta muodostuneesta klusterista:

> *Aineistosta tehtiin havainto, että opettajat kertoivat millaisia työtapoja ovat käyttäneet tai millaisiin työtapoihin ovat ohjanneet oppilaitaan ohjelmointiopetuksen*

> *yhteydessä. Nämä työtavat jakautuivat teoriasta löytyviin testaamiseen ja virheenkorjaamiseen, kokeilemiseen ja iterointiin sekä uudelleenkäyttöön ja uudelleenversiointiin.*

Esimerkki sisällön pohjalta muodostuneesta klusterista:

> *Aineistosta tehtiin havainto, että opettajat kertoivat työtapoja kysyttäessä sen, millaisissa ohjelmointiympäristöissä he ovat opettaneet ohjelmointia. Nämä ohjelmointiympäristöt jaettiin koneettomiin ohjelmointiympäristöihin ja tietokoneohjelmointiympäristöihin. Tietokoneohjelmointiympäristö jakautui vielä pienempiin luokkiin, joita olivat näytöllä tapahtuva ohjelmointi, robotiikka ja mikrokontrollerit. Nämä voitiin jakaa vielä sovelluksen tai käytetyn laitteen mukaan.*

Pelkistettyä ja ryhmiteltyä aineistoa analysoitiin ja niiden avulla voitiin muodostaa lopulta yläluokat. Tätä vaihetta kutsutaan abstrahoinniksi (Tuomi & Sarajärvi, 2018). Nämä yläluokat jaoteltiin teorian mukaisiin ohjelmoinnin opetuksen sisältöihin, jotka olivat ohjelmoinnin opetuksen käsitteet, ohjelmoinnin opetuksen käytänteet ja ohjelmoinnin opetuksen päämäärät. Yläluokiksi muodostuivat ohjelmoinnin opetuksen käsitteiden kohdalla ohjelmointikielten ja -ympäristöjen käsitteet sekä ohjelmoinnin käsitteet.

Ohjelmoinnin *opetuksen käytänteiden* kohdalla yläluokiksi muodostuivat ohjelmointitehtävien lähestyminen opettajan toimesta, ohjelmoinnin opetuksen työskentelymuodot, tehtävätyypit ohjelmoinnin opetuksessa ja oppilaan ohjelmoinnin oppimisen strategiat. Oppilaan ohjelmoinnin strategiat jaettiin teorian (ks. Brennan & Resnick, 2012) mukaan *testaamiseen ja virheenkorjaamiseen, muokkaamiseen ja uudelleenkäyttämiseen* ja kolmantena *kokeilemiseen ja iterointiin*. Vastaukset sisälsivät mainintoja eri sovelluksista ja laitteista, joita oli käytetty osana ohjelmoinnin opetusta. Nämä jaoteltiin *koneettomaan ohjelmointiin* ja *tietokoneohjelmointiin* sen mukaan, missä tai millä laitteilla ohjelmoitiin. Tietokoneohjelmointi jaettiin neljään teemaan, jotka olivat *näytöllä tapahtuva ohjelmointi*, *robotiikka*, *mikrokontrollerit* ja *lisättyyn todellisuuteen (eng. augmented reality)*. Tietokoneohjelmoinnin neljä teemaa ja niiden sisällöt on esitelty taulukossa 7. Lisäksi tehtävätyypit jaoteltiin kolmeen kategoriaan, jotka olivat *valmiit materiaalipaketit*, *opettajan itse valmistelemat materiaalit* ja *oppilaan omat projektit*.

Ohjelmoinnin *opetuksen päämäärien* kohdalla yläluokiksi muodostuivat monilukutaito, ajattelun taidot, työ- ja elinkeinoelämä, elämänhallintataidot sekä ohjelmointitaito. Näiden edellä mainittujen analyysivaiheiden lopuksi aineistoa tarkasteltiin apukysymysten avulla, jonka jälkeen voitiin muodostaa tutkimuksen tulokset. Tutkimuksen tuloksiin liitettiin suoria lainauksia aineistosta kuvaamaan saatuja tuloksia. Lopuksi tutkimuksen tuloksista voitiin muodostaa johtopäätökset. Taulukossa 5 on esitelty aineistosta löydetyt ohjelmoinnin opetuksen yläluokat.

TAULUKKO 5. Analyysin avulla muodostetut yläluokat luokiteltuna teorian pohjautuviin yläluokkiin

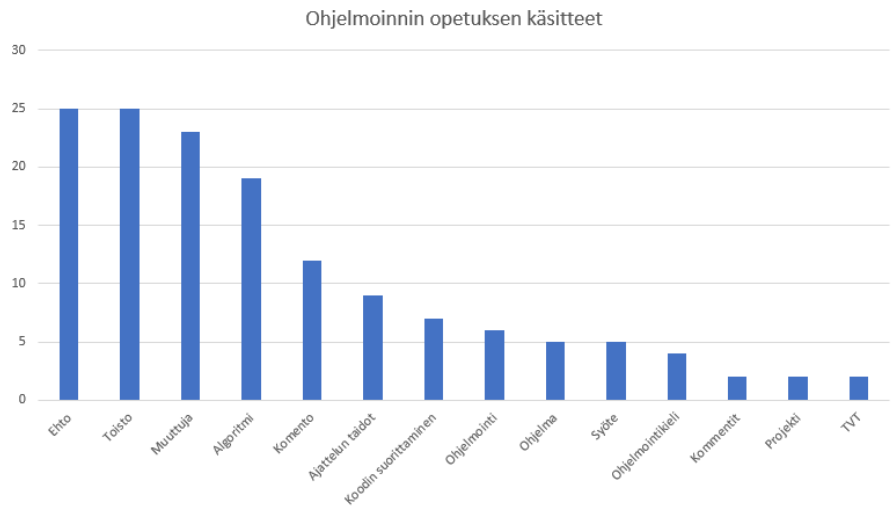
Käsitteet	**Käytänteet**	**Päämäärät**
Ohjelmointikielten ja -ympäristöjen käsitteet	Opettajan toiminta	Monilukutaito
	Työskentelymuodot	Ajattelun taidot
Ohjelmoinnin käsitteet	Tehtävätyypit	Työ- ja elinkeinoelämä
	Oppimisen strategiat	Elämänhallintataidot
		Ohjelmointitaidot

6. Tutkimuksen tulokset

Tässä luvussa esitellään tutkimuksen analyysin tulokset. Tulokset jaetaan kolmeen alalukuun tutkimuskysymysten mukaan, joista ensimmäisessä alaluvussa tarkastellaan ohjelmoinnin opetuksessa korostuvia käsitteitä. Toisessa alaluvussa tarkastellaan ohjelmoinnin opetuksen käytänteitä eli siinä esiintyviä tehtäviä ja työtapoja. Kolmannessa alaluvussa tarkastellaan ohjelmoinnin opetuksen päämääriä eli opetuksen tavoitteita ja ohjelmointiosaamisen vaikutuksia oppilaan tulevaisuudelle. Tuloksilla pyritään vastaamaan pääkysymykseen, *millaista ohjelmoinnin opetus on suomalaisissa peruskouluissa*?

6.1 Mitkä ohjelmointiin liittyvät käsitteet korostuvat ohjelmoinnin opetuksessa?

Vastaajilta kysyttiin ”*mitä ohjelmointiin liittyviä käsitteitä pidät tärkeimpinä ohjelmoinnin opetuksessa?”*. Tarkentavaksi ohjeeksi annettiin mainita enintään viisi omasta mielestä tärkeintä käsitettä. Vastauksissa *(n=42)* kolme eniten mainintoja saanutta käsitettä olivat *ehto (25)*, *toisto (25)* ja *muuttuja (23)*. Muita tilastollisesti merkittäviä käsitteitä vastausten perusteella olivat *algoritmi (19)* ja *komento (12)*. *taitoihin* liittyviä käsitteitä sisältäneitä vastauksia antoi yhdeksän vastaajaa, eli noin joka viides vastaajista. Algoritmi -käsitteen alle sidottiin yhteen käsitteet *algoritmi (9), koodi (6)* sekä *funktio (5)*. Vähäisiä mainintoja *(n=2–7)* saivat myös käsitteet liittyen *koodin suorittamiseen*, *ohjelmointiin*, *ohjelmaan*, *syötteeseen*, *ohjelmointikieleen*, *kommentteihin*, *projekteihin* sekä *tieto- ja viestintätekniikkaan*.



KUVIO 4. Ohjelmoinnin opetuksen käsitteiden jakautuminen vastauksissa

TAULUKKO 6. Ohjelmoinnin opetuksen käsitteiden jakautuminen luokka-asteen mukaan

Käsite	**1.–2.**	**3.–6.**	**7.–9.**	**Muu**	**Yht.**
Ehto	1	5	17	2	25
Toisto		6	17	2	25
Muuttuja	1	4	16	2	23
Algoritmi	3	4	10	2	19
Komento	4	7	1		12
Ajattelun taidot	1	3	5		9
Koodin suorittaminen	1	1	4	1	7
Ohjelmointi	2	3	1		6
Ohjelma	1	2	2		5
Syöte			5		5
Ohjelmointikieli	1	3			4
Kommentit			2		2
Projekti		1	1		2
TVT		1	1		2

Taulukon 6 avulla voidaan havaita kokonaistuloksissa eniten mainintoja saaneiden käsitteiden *ehto*, *toisto* ja *muuttuja* olevan tärkeässä roolissa erityisesti luokka-asteiden 7.–9. opetuksessa, ja tulleen usein mainituksi jo 3.–6. luokka-asteen opetuksessa. Luokka-asteilla 1.–2. kyseiset käsitteet eivät saavuta vastaavaa merkittävyyttä. Koska vastaajista kuitenkin vain neljä ilmoitti opettavansa 1.–2. luokka-astetta (ks. taulukko 3), ei tuloksia tämän ja muiden luokka-asteiden välillä ole mielekästä vertailla.

KUVIO 5. Ohjelmoinnin opetuksen käsitteitä ja termejä ennen klusterointia Nvivo-analyysisovelluksen muodostamana sanapilvenä. Keskellä punaisella eniten mainintoja saaneita käsitteitä ja termejä.

6.2 Millaisten tehtävien ja työtapojen avulla ohjelmointia opetetaan?

Vastaajilta kysyttiin kysymykset ”*millaisten tehtävien avulla olet opettanut ohjelmointia?”* ja ”*millaisia työtapoja olet käyttänyt opettaessasi ohjelmointia?”*. Kummallakin kysymyksellä pyrittiin löytämään vastauksia samaan tutkimuskysymykseen, joten kysymyksiin saadut vastaukset yhdistettiin aineiston analyysissä. Saaduista vastauksista muodostetut yläkäsitteet ovat *ohjelmointitehtävien lähestyminen opettajan toimesta*, *ohjelmoinnin opetuksen työskentelymuodot*, *tehtävätyypit ohjelmoinnin opetuksessa* ja *oppilaan ohjelmoinnin oppimisen strategiat*.

6.2.1 Ohjelmointitehtävien lähestyminen opettajan toimesta

Opettajien rooli tehtäviä lähestyttäessä oli vaihdellut esimerkin antamisen, ongelmalähtöisen tehtävänannon asettamisen, valmiin materiaalin käytön tai yhteisten suunnittelutuokioiden vä-

lillä. Ohjelmointiopetuksen sisältöjä ja tehtäviä oli lähestytty oppilaiden kanssa yhteisesti erilaisten esimerkkien kautta. Vastaajat olivat antaneet oppilaille mallikoodeja tai tehtävien mallipohjia, joista oppilaat etenisivät itsenäisiin tai soveltaviin tehtäviin. Vastaajat kertoivat myös näyttäneensä opettajajohtoisesti esimerkkejä koodien rakentamisesta, laitteen tai ympäristön perustoiminnoista sekä tulevista tehtävistä. Esimerkkien avulla tutustuttiin sekä keskeisiin että uusiin käsitteisiin. Syiksi esimerkkien käyttöön vastaajat mainitsivat rajallisen ajan sekä alkuun pääsemisen helpottamisen. Vastaajat 35 ja 41 kuvasivat toimintaansa ohjelmointitehtävien lähestymisessä seuraavilla tavoilla:

> *”Esimerkkejä ja sanallisia tehtävänantoja. Rajallisen ajan vuoksi alkuun valmiita esimerkkejä, joista sitten jatketaan.” (Vastaaja 35, aineenopettaja)*

> *”Opettajajohtoinen esittely tai johdatus aiheeseen, jossa opitaan pari uutta käsitettä tai merkintää” (Vastaaja 41, aineenopettaja)*

Osa vastaajista kertoi lähestyneensä ohjelmointitehtäviä ongelmalähtöisen tehtävänasettelun kautta. Oppilaat johdatettiin ohjelmointitehtäviin erilaisten haasteiden, projektien tai tutkimustehtävien avulla. Esimerkkeinä vastaajat mainitsivat annetuista ongelmalähtöisistä tehtävänasetteluista esimerkiksi robotin, jonka täytyi suorittaa jokin annettu haaste. Yksi vastaaja kertoi näyttäneensä myös videon robotin toiminnasta, ja oppilaan tehtävänä olisi ohjelmoida oma robottinsa toistamaan esimerkkivideon robotin toiminta. Lähtökohtana tehtävänannoissa oli siis ratkaista jokin haaste, ongelma tai pulma. Ongelmalähtöisiä tehtävänantoja kuvailtiin esimerkiksi tällä tavoin:

> *”Ongelmanratkaisu, tutkimustehtävät ja projektit. Pyrin antamaan oppilaille tutkimiseen ja kokeiluun perustuvia tehtäviä, jolloin he pääsevät tekemään itse oppimisen kannalta olennaisia johtopäätöksiä.” (Vastaaja 19, pedagoginen asiantuntija)*

> *”Mallivideo robotin toiminnasta, tehtävänä rakentaa koodi, joka mahdollistaa toiminnan” (Vastaaja 33, luokanopettaja)*

Ohjelmoinnin opetuksessa oli käytetty myös valmista materiaalia. Osa valmiista materiaalista oli kolmannen osapuolen tuottamaa, kun taas osa materiaalista oli vastaajien itsensä tuottamaa.

Osa valmiista materiaalista antoi palautetta itsenäisesti, ja vapautti opettajalle näin aikaa esimerkiksi auttaa niitä oppilaita, jotka kaipasivat enemmän tukea. Valmiissa materiaaleissa oli mukana erilaisia vihjeitä tehtäviin ja esimerkkikoodeja.

> *"...valmiit alustat, joissa palaute tulee omaan tahtiin ilman open työpanosta" (V7, aineenopettaja)*

> *"Itsenäinen eteneminen tekemäni materiaalin mukaan, jolloin opettajalla aikaa auttaa kädestä pitäen heikoimpia" (Vastaaja 18, aineenopettaja)*

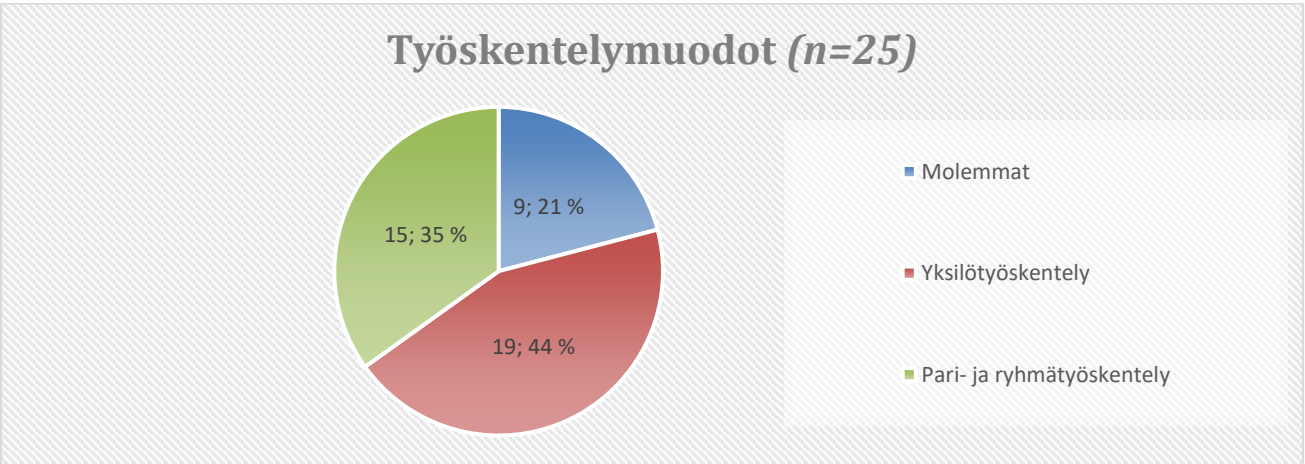
Jotkut vastaajista lähestyivät ohjelmoinnin opetusta suunnitelmallisuuden kautta. Apuna käytettiin pseudokoodin kirjoittamista tai vuokaavioiden piirtämistä, joilla hahmotettiin ohjelman algoritmeja ja toimintalogiikkaa sekä kartoitettiin erilaisia vaihtoehtoja.

> *"Vuokaavio: opitaan ensin suunnitelmallisuutta, loogisuutta ja kaikkien vaihtoehtojen kartottamista" (Vastaaja 20, aineenopettaja)*

> *"Först skriva pseudokod, sen programmera" (Vastaaja 38, aineenopettaja)*

6.2.2 Ohjelmoinnin opetuksen työskentelymuodot

Kysyttäessä ohjelmoinnin opetuksen työtapoja, yli puolet vastaajista *(n=25)* vastaajista mainitsivat yksilö- *(n=19)* tai pari- ja ryhmätyöskentelyn *(n=15)*. Osa vastaajista (ks. kuvio 6) mainitsi kaikki edellä mainitut työskentelymuodot *(n=9)*.



KUVIO 6. Ohjelmoinnin opetuksen työskentelymuotojen maininnat

Yksilötyöskentelyn maininneet kokivat sen mahdollistavan tehokkaamman osallistumisen ja motivoivan niitä oppilaita, jotka pitävät itsenäisestä työskentelystä. Lisäksi yksilötyöskentelyn koettiin mahdollistavan tehokkaamman eriyttämisen, kun oppilaat saavat edetä valmiiden materiaalien ja tutoriaalien avulla itsenäisesti, jolloin myös opettajalle jää enemmän aikaa auttaa apua tarvitsevia oppilaita. Yksilötyöskentelyä kuvattiin ja perusteltiin vastaajien 28 ja 32 toimesta:

> *”Opetan suurta ja heterogeenistä luokkaa, jossa on myös erityisoppilaita. Olen tehnyt opetusvideoita scratchistä. Jokainen etenee omaa vauhtiaan; jokaisella kone ja kuulokkeet. Tunti alkaa yhteisellä (helpolla) opetusosuudella, sen jälkeen kukin etenee opetusvideoiden avulla omaa vauhtiaan ja minulla on aikaa auttaa ongelmissa ja debuggaamisessa” (Vastaaja 28, luokanopettaja)*

> *”Jotkut tykkäävät tosi paljon, kun saavat itsekseen pähkäillä koodausjuttuja” (Vastaaja 32, luokanopettaja)*

Pari- tai ryhmätyöskentelyn maininneet kokivat sen tasoittavan oppilaiden tasoeroja. Sen nähtiin tarjoavan keinot yhteis- ja vertaisoppimiselle, jolloin oppilaat pääsivät jakamaan ajatuksiaan ja ratkomaan ongelmia yhdessä kaverin kanssa. Yhteistyön koettiin tuovan voimavaroja, joiden nähtiin edesauttavan ohjelmoinnin oppimista, esimerkiksi käsitteiden osalta. Osana pari- ja ryhmätyöskentelyä opeteltiin myös lähdekoodin kommentointia, jonka nähtiin olevan yhteydessä hyviin ohjelmointikäytäntöihin. Parityöskentelyn nähtiin olevan myös ehtona toisen oppilaan ohjelmointiin koneettoman ohjelmoinnin harjoituksissa. Pari- ja ryhmätyöskentelyn yhteydessä mainittiin myös tehostettu kisällioppiminen osana ohjelmoinnin opetusta.

> *”Pari- ja ryhmätyötä tukevat käsitteiden oppimista, koska tehdään yhteistyötä. Suurempi työ voidaan myös jakaa osiin. Kaikkien taidot saadaan käyttöön. Toisen oppilaan vahvuus voi olla koodaus, toisen esim. kuvankäsittely. Ryhmätyössä kaikkien heikotkin ominaisuudet voivat vahvistua” (Vastaaja 14, aineenopettaja)*

> *”Toinen malli on ollut se, jossa oppilaat luovat itse koodeja esim. kirjoittamalla ne vihkoon ja sitten toinen oppilas toimii koodin mukana. Tämä on hyvin havainnollistava, helppo ja yksinkertainen tapa opettaa koodausta ja se sopii nuorille oppilaille hyvin” (Vastaaja 21, erityisluokanopettaja)*

6.2.3 Tehtävätyypit ohjelmoinnin opetuksessa

Kysyttäessä ohjelmoinnin opetuksessa vastaajien käyttämistä tehtävistä, vastaajat mainitsivat useita eri sovelluksia ja laitteita, joita olivat käyttäneet osana ohjelmoinnin opetusta. Vastaukset voitiin jaotella *koneettomaan ohjelmointiin* ja *tietokoneohjelmointiin*.

Vastaajat kertoivat käyttäneensä *koneetonta ohjelmointia* esimerkiksi ohjelmoinnin peruskäsitteiden opettamiseen. Koneettomissa ohjelmointiharjoituksissa oppilaat muodostivat tarkkoja toimintaohjeita arkisiin toimenpiteisiin, kuten leipomiseen tai aamutoimiin. Oppilaat ohjelmoivat myös toisiaan laatimalla kirjallisia toimintaohjeita toistensa suoritettaviksi. Tällaiset harjoitukset koettiin havainnollistaviksi ja ymmärrystä lisääviksi. Koneetonta ohjelmointia harjoiteltiin myös erilaisten konkreettisten leikkien ja piirtotehtävien avulla, joista vastaajat antoivat muun muassa seuraavanlaisia esimerkkejä:

> ”*Keksitään tarkkoja ohjeita erilaisiin tilanteisiin (esim. leipominen tms.)” (Vastaaja 4, luokanopettaja)*
>
> *”Oppilaat ovat luoneet komentoketjuja, joita he testaavat toisilla oppilailla käytännössä. Tämä on hyvin havainnollistavaa ja saa heidät ymmärtämään, mitä koodaus käytännössä merkitsee” (Vastaaja 21, erityisluokanopettaja)*
>
> *”Kynällä ja paperilla ruudukolla esim. aarteenetsintää (kulje reitti)” (Vastaaja 36, luokanopettaja)*

Tietokoneohjelmointia vastaajat olivat käyttäneet osana ohjelmoinnin opetusta hyödyntäen useita eri ohjelmointiympäristöjä ja sovelluksia. Nämä tulokset on esitelty tiivistetysti taulukossa 7. Vastaajat antoivat esimerkkejä tai perusteluita tietokoneohjelmoinnin näytöllä tapahtuvista ohjelmointitehtävistä, joissa käytettiin Scratchia, Racketia, Code.orgia, JavaScriptia ja Python Repl.itiä.

Scratchilla toteutetut ohjelmointitehtävät pitivät sisällään erilaisia pelejä, sovelluksia sekä animaatioita. Scratchilla harjoiteltiin komentosarjoja ja harjoiteltiin koodauksen perusteita, mutta sitä käytettiin myös projektinomaisissa ohjelmointitehtävissä. Scratch koettiin monipuoliseksi ohjelmointiympäristöksi, joka mahdollisti luovien ja kokeilevien tehtävätyyppien käytön.

Scratchilla toteutettavia ohjelmointiprojekteja integroitiin esimerkiksi niissä käsiteltävien teemojen kautta muihin oppiaineisiin:

> *”Scratch on erittäin monipuolinen ja haastava työkalu – Parasta on, jos on joku projekti menossa, missä jotain peliä tai ohjelmaa tarvitaan. Katsomme esimerkkejä, remixataan, yritetään, testataan, vertaillaan ratkaisuja, annetaan parannusvinkkejä jne. Hiotaan peli paremmaksi jne.” (Vastaaja 1, luokanopettaja)*

> *”Scratch Jr, Beebot ja WeDo alkuopetuksessa, alkeet konkretisoituvat” (Vastaaja 24, luokanopettaja)*

> *”Oppilaat ovat laatineet ensin mallin mukaan komentosarjoja esim. Scratchilla ja sitten tehneet jonkin tehtävän, vaikkapa jouluanimaation, tutkien Scratchin ominaisuuksia” (Vastaaja 25, luokanopettaja)*

Racketilla harjoiteltiin ohjelmointia piirtäen geometrisiä kuvioita. Jotkut vastaajat kertoivat integroineensa Racketilla toteutettavia tehtävätyyppejä suoraan osaksi matematiikan opetusta, ja siihen löytyi tehtäviä esimerkiksi Koodiaapisesta ja MAOL-materiaaleista.

> *”Teemme monikulmioita kilpikonnagrafiikalla Racketilla, testataan tuleeko haluttu monikulmio, korjataan koodia halutun tuloksen saamiseksi, käytetään ja muokataan tehtyä koodia uudelleen seuraavan monikulmion piirtämiseen.” (Vastaaja 8, aineenopettaja)*

Vastaajat antoivat muutamia esimerkkejä myös JavaScriptistä, jolla ohjelmoitiin esimerkiksi laskin, Code.orgista, johon löytyi valmiita opetuspaketteja, sekä Python Repl.itistä, jolla luotiin osallistava novelli ohjelmoinnin avulla. OpenProcessingia oli käytetty kuvataiteen töiden koodaamiseen. Muutamia mainintoja käytetyistä tehtävätyypeistä tai ohjelmointiympäristöistä saivat myös Python, Khan Academy, C++ ja Lightbot.

> *”Joitakin ohjaan ensin tekemään code.org -tai muiden ”valmiiden opetuspakettien” perusopetuskierrokset ja sitten vasta luovempia tehtäviä” (Vastaaja 1, luokanopettaja)*

> *”Olemme kirjoittaneet osallistavan novellin choose your own adventure -tapaan, eli lukijan ratkaisut vaikuttavat tarinan kulkuun. Toteutimme Pythonilla Repl.it-alustalla” (Vastaaja 26, aineenopettaja)*

Tietokoneohjelmointia harjoiteltiin myös hyödyntäen robotiikkaa. Vastaajat antoivat esimerkkejä tai perusteluita tietokoneohjelmoinnin robotiikan ohjelmointitehtävistä, joissa käytettiin Legoja, VEXejä sekä Blue-/Beebotteja.

Vastaajat kertoivat käyttäneensä Legoja sekä VEX-rakennussarjoja ohjelmointitehtävissä, joissa laite tai robotti haluttiin saada toimimaan halutulla tavalla. Tehtävätyypeissä hyödynnettiin eri antureita ja harjoiteltiin myös mekaniikkaa robottien rakentamisen avulla. Vastaajat kokivat robotiikkaa hyödyntävien tehtävätyyppien mahdollistavan monipuoliset ja skaalautuvat ohjelmointiprojektit. Vastaajat kertoivat myös Legoja ja VEX-rakennussarjoja käytettävän osana erilaisia kilpailuja, joiden tehtävät vaihtelivat avoimista tehtävänannoista tarkkaan määriteltyihin tehtäviin. Vastaajat antoivat seuraavanlaisia esimerkkejä Legojen ja VEX-rakennussarjojen hyödyntämisestä osana ohjelmointiopetusta:

> *”Opetan ohjelmointia sellaisissa ympäristöissä, joilla saadaan jokin laite toimimaan halutulla tavalla, esim. Lego mindstorms ja microbit. Näissä yleensä on mukana jokin sensori, joka tarkkailee ympäristöä ja tietyssä tilanteessa ehtojen täyttyessä jotain muuta konkreettista tapahtuu.” (Vastaaja 3, luokanopettaja)*

> *”Alkuopetuksessa olemme käyttäneet lego wedo-robotteja niin, että kokosimme ne ohjeen mukaan. Sen jälkeen niitä ohjelmoitiin ja testailtiin.” (Vastaaja 32, luokanopettaja)*

> *”Lego ja VEX- robotiikkarakentelu ja ohjelmointi kisoja varten: Monipuolisia ja skaalautuvia projekteja.” (Vastaaja 43, aineenopettaja)*

Blue- ja Beebotteja käytettiin vastaajien mukaan usein osana alkuopetuksessa tapahtuvaa ohjelmoinnin opetusta. Niiden avulla harjoiteltiin liikeohjelmien antamista lyhyiden komentosarjojen avulla. Vastaajat kertoivat hyödyntäneensä Blue- ja Beebottien ohjelmointitehtäviä integroiden niitä osaksi muuta opetusta:

> ”*Bluebotteja/beebotteja olemme ohjelmoineet aika paljonkin monissa eri oppiaineiden tehtävissä. (Esim. Matikassa nostetaan kortti jossa lasku, sitten ohjelmoidaan bluebot menemään oikean vastauksen luo) Pienten kanssa on tärkeää jatkuva testaaminen ja lyhyet komentosarjat, jotta ymmärrys pysyy mukana. Mikäli robotti ei mene haluttuun paikkaan, koodi mietitään uudestaan.” (Vastaaja 32, luokanopettaja)*

Tietokoneohjelmointia harjoiteltiin lisäksi mikrokontrollereita ohjelmoimalla. Vastaajat antoivat esimerkkejä tai perusteluita tietokoneohjelmoinnin mikrokontrollereiden ohjelmointitehtävistä, joissa käytettiin Micro:bittejä tai muita ohjattavia mikropiirejä.

Micro:bittejä ja muita mikrokontrollereita käytettiin vastaajien mukaan erilaisten todenpnäköisyyteen liittyvien pelien tekemiseen ja koodaamisen perusteiden opiskeluun, sekä erilaisten laitteiden, kuten ledimainosten ja laserharjoitusaseiden ohjelmointiin. Vastaajien mukaan Micro:bittejä hyödyntävissä ohjelmointitehtävissä käytettiin myös erilaisia antureita. Taulukossa 7 on kuvattu tietokoneohjelmoinnin tehtävätyypeissä mainitut ohjelmointiympäristöt jaoteltuna neljään eri tyyppiin käyttöliittymän mukaan, sekä annettuja esimerkkejä niissä käytetyistä tehtävätyypeistä.

> *”Todennäköisyyspelejä (kolikonheitto, KPS, noppa) ollaan tehty Micro:biteillä, testataan toimiiko ja korjataan virheitä, edetään seuraavaan haasteeseen käyttäen edellistä koodia hyödyksi.” (Vastaaja 8, aineenopettaja)*

> *”Esimerkiksi ohjelmoitavan mikropiirin avulla tehty laitteita (ledimainokset, laserharjotusase), jos koodi on oikeanlainen, laite toimii halutusta eli helppo testaus” (Vastaaja 40, aineenopettaja)*

TAULUKKO 7. Tietokoneohjelmoinnissa käytettyjä ohjelmointiympäristöjä, sovelluksia ja tehtävätyyppejä

Missä?	Ohjelmointiympäristö/laite	Mitä tehty?
Näytöllä tapahtuva	Scratch & Scratch Jr.	• Ohjelmointiprojektit • Tarinalliset ja teemalliset animaatiot • Pelit • Ohjelmat • Remixaus • Koodauksen perusteet • Scratch Jr. alkuopetuksessa
	Racket	• Geometristen kuvioiden piirtäminen • Integrointi matematiikkaan
	Code.org	• Valmiit opetuspaketit • Pelit
	JavaScript	• Laskimen ohjelmointi
	HTML	• Lähdekoodin remixaaminen
	MIT App Inventor	• Valmiit tutoriaalit
	CodeCombat	• Pelit
	OpenProcessing	• Kuvataiteen töiden prosessointi
	Python Repl.it	• Osallistava novelli
Robotiikka	Lego (EV3, WeDo)	• Kilpailut • Monipuoliset projektit • Antureiden käyttö • Mekaniikka • Konkretia ja kokeilu • WeDo alkuopetuksessa
	Beebot / Bluebot	• Liikekomentojen harjoittelu • Alkuopetus • Integrointi eri oppiaineisiin (esim. äidinkielessä alkutavulta lopputavulle)
	Vex	• Kilpailut • Monipuoliset projektit • Mekaniikka
Mikrokontrollerit	Micro:bit	• Todennäköisyyteen perustuvat projektit (kolikonheitto, kivi-paperi-sakset, noppa) • Elektroniikkaharjoitukset
	Muut mikrokontrollerit	• Ledimainokset • Laserharjoitusase
Lisätty todellisuus	Osmo	• Ohjelmointipelit

Tehtävätyypit voitiin jakaa vielä kolmeen osaan opetusmateriaalin tuottajan mukaan. Osiot ovat *valmiit materiaalit*, *opettajan itse tekemät materiaalit* sekä *materiaalittomat tehtävät* eli oppilaan omat projektit.

Valmiilla materiaaleilla tarkoitetaan tässä yhteydessä kolmannen osapuolen luomia valmiita opetusmateriaaleja tai tehtävänantoja. Vastaajat olivat käyttäneet osana ohjelmoinnin opetusta valmiita opetusmateriaaleja, joita olivat esimerkiksi erilaiset koodaustehtävät, pelit, itse palautetta antavat ohjelmointialustat, tutoriaalit, videot, oppikirjojen tehtävät sekä ohjelmointikurssit. Vastaajat kertoivat käyttäneensä valmiita materiaaleja Code.orgilta, Micro:bitiltä, MIT App Inventorilta, Khan Academystä, Tie koodariksi -julkaisusta, MAOL-materiaaleista, LUMA-keskukselta, FiTech 101:sta, SanomaProlta sekä Innokas -verkoston materiaaleista. Valmiiden materiaalien käyttöä vastaajat perustelivat niiden helppouden ja opettajalle jäävän ylimääräisen auttamisajan perusteella. Lisäksi oman materiaalin tuottaminen koettiin työlääksi:

> *”En ole tehnyt omia tehtäviä, se on tuntunut aivan liian työläältä.” (Vastaaja 41, aineenopettaja)*

Opettajien itse tekemät materiaalit pitivät sisällään esimerkiksi opetusvideoita, tutkimiseen ja kokeiluun perustuvia tehtäviä, ongelmalähtöisiä tehtävänantoja sekä muita pieniä tehtäviä. Tällaisia opettajien itse tekemiä materiaaleja tai tehtävänantoja olivat esimerkiksi erilaiset budjettilaskut, pelit, urheilukisojen sarjataulukointi, laskimen ohjelmointi, kalenterin tekeminen, robottihaasteet ja osallistavat novellit.

Materiaalittomat tehtävät eli oppilaiden omat projektit nousivat vastaajien mukaan usein erilaisten haasteiden tai kilpailutehtävien muodossa. Esimerkiksi robotiikkakisat antoivat oppilaille tehtävän, jonka ratkaisutyyli oli hyvin vapaa eli oppilaat saivat itse ratkaista tehtävän omaa luovuuttaan hyödyntäen. Oppilaiden omista projekteista mainittiin myös omat tarinat, pelit ja animaatiot, joihin usein liittyi jokin annettu teema.

6.2.4 Oppilaan ohjelmoinnin oppimisen strategiat

Vastaajat kertoivat erilaisista tehtävistä ja työtavoista, joita he käyttivät opetuksessaan ja joita oppilaat käyttivät ratkaistakseen opettajan antamia tehtäviä. Teoriaan nojaten aineistosta pys-

tyttiin määrittelemään kolme erilaista ohjelmoinnille ominaista oppimisen strategiaa: *testaaminen ja virheenkorjaus (n=24), muokkaaminen ja uudelleenkäyttäminen (n=19)* ja kolmantena *kokeileminen ja iterointi (n=14)*. Teorian mukaista (ks. 3.3. Ohjelmointi ja ohjelmoinnillisen ajattelun yhteys) neljättä teemaa *abstrahointia ja modulointia* ei aineistossa kuvattu. Kun tutkittiin tyylejä, joita oppilaat käyttivät opettajan antamissa tehtävissä, 36 vastaajaa kuvasi jotain näistä oppimisen strategioista. Viisi vastaajista kuvasi vastauksessaan kaikkia näitä oppimisen strategioita.

> *”Opetan ohjelmointia sellaisissa ympäristöissä, joilla saadaan jokin laite toimimaan halutulla tavalla, esim. Lego mindstorms ja microbit. Näissä yleensä on mukana jokin sensori, joka tarkkailee ympäristöä ja tietyssä tilanteessa ehtojen täyttyessä jotain muuta konkreettista tapahtuu. Tällaisissa tehtävissä korostuvat testaaminen ja virheiden korjailu.” (Vastaaja 3, luokanopettaja)*

> *”Beebottien ohjelmointia alkutavulta lopputavulle. Yrityksen ja erehdyksen kautta, tyyliin: ohjelmoi, kokeile, menikö oikein, korjaa, yritä uudestaan.” (Vastaaja 39, luokanopettaja)*

Vastaajien mukaan *testaamista ja virheenkorjaamista* tapahtui luonnollisesti osana eri ohjelmointitehtäviä. Vastaajien mukaan koodia kirjoittaessa virheitä syntyy aina, jopa esimerkkikoodia kirjoittaessa, ja niitä etsitään ja korjataan. Ohjelmoidessaan oppilaat testailevat ja korjailevat omaa projektiaan jatkuvasti. Testaamisen ja virheen korjaamisen avulla oppilaat pyrkivät kehittämään koodiaan siihen suuntaan, että se toimii halutulla tavalla. Oppilaat etsivät virheitä myös toistensa projekteista. Vastaajat kertoivat myös käyttäneensä tehtävätyyppejä, joissa oppilaan tehtävänä oli löytää ja korjata virheitä valmiista koodista. Vastaajien mukaan konkretia, eli esimerkiksi robottien tai muiden konkreettisten laitteiden käyttö osana opetusta, helpottaa testaamista ja virheiden löytämistä. Vastaajat antoivat esimerkkejä testaamisesta ja virheenkorjaamisesta:

> *”Olen antanut oppilaille koodipohjan ylläolevaan tehtävään, ja olemme käyneet läpi miten koodi toimii ja mitä tehtävällä tavoitellaan. Koodipohjaa on jokainen voinut sitten muokata enemmän tai vähemmän kirjoittaessaan osallistavaa novellia.” (Vastaaja 26, aineenopettaja)*

”Kommentoitu esimerkkisivu html-koodia. Tutustu ja muokkaa. Vaihda kaikki tekstit ja kuvat. Remixaus tärkeä osa.” (Vastaaja 33, luokanopettaja)

”Valmiin koodin kopioiminen ja testaaminen, jotta jokainen kokisi saavansa jotain aikaan ilman että pitää nähdä kauhean paljon vaivaa.” (Vastaaja 29, aineenopettaja)

Vastaajien mukaan *muokkaamista ja uudelleenkäyttämistä* tapahtui osana ohjelmoinnin opetusta, kun tehtävänantona oli esimerkiksi kopioida mallikoodi ja muokata sitä omaan käyttöön soveltaen. Vastaajien mukaan tällaisissa tehtävänannoissa oppilaille annettiin valmis mallikoodi tai valmiita esimerkkejä koodien osista, joita oppilaan tuli hyödyntää tai uudelleen versioida osana omaa työtään. Joissain tehtävätyypeissä myös pelkällä parametrien muokkaamisella saatiin jotain uutta aikaan. Myös omien aiempien projektien koodeja hyödynnettiin uudestaan osana uuden luomista. Joidenkin ohjelmointiympäristöjen, kuten Scratchin, mainittiin tarjoavan helpon tavan uudelleen versioida myös muiden käyttäjien tekemiä projekteja erillisen ”remix” näppäimen avulla. Uudelleenkäyttämisen ja muokkaamisen nähtiin motivoivan oppilaita ohjelmoimaan, kun jokainen koki saaneensa helposti aikaan jotain valmista.

”Olen esim JavaScriptiä käyttäen teettänyt erilaisia tehtäviä, joissa ensin käyttäjiltä kysytään asioita ja sen jälkeen toimitaan. Esim ohjelmoi laskin.” (Vastaaja 17, aineenopettaja)

Vastaajien mukaan *kokeilemista ja iterointi* tapahtui osana oppilaiden omia ohjelmointiprojekteja. Oppilaat testailivat ja kehittelivät omia ohjelmointiprojektejaan muilta saatujen vinkkien ja ideoiden, oman uuden oppimisen, omien ajatusten ja näkemysten sekä erilaisten ongelmalähtöisten tehtävänantojen perusteella. Myös oppilaiden keskinäinen projektien vertailu nähtiin vastaajien mukaan olevan yhteydessä oppilaiden omien projektien kehittymiseen. Oppiessaan uutta ohjelmoinnista tai ohjelmointiympäristöstä, oppilaat saivat vastaajien mukaan uusia keinoja ja näkemyksiä kehittää omia projektejaan.

Tutkimiseen ja kokeiluun perustuvat tehtävänannot nähtiin vastaajien mukaan olevan tärkeitä ohjelmoinnin oppimisessa. Jotkut vastaajista olivat myös käyttäneet vuokaavioita tai kirjoituttaneet oppilailla pseudokoodia harjoituttaakseen oppilaita hahmottamaan koodin eri mahdollisuuksia ja keinoja toteuttaa projekteja teorian tasolla. Erilaiset kilpailuasetelmat tai ongelma-

lähtöiset tehtävänannot toimivat vastaajien mukaan myös innoittajana tuotekehittelylle, eli iteroinnille. Jos tehtävänannon aiheena oli kehittää laite suorittamaan tietty tehtävä, mutta tehtävänanto oli muuten avoin, ajoi se oppilaat suunnittelemaan, keksimään ja kehittelemään omaa työtään päämääränään saada aikaan oman projektinsa ominaisuuksien jatkuva parantelu.

6.3 Millaisia näkemyksiä opettajilla on ohjelmoinnin opetuksen keskeisistä tavoitteista ja vaikutuksista oppilaiden tulevaisuudelle?

Vastaajilta kysyttiin kysymykset ”*mitkä ovat mielestäsi keskeiset tavoitteet, joiden vuoksi ohjelmoinnin opettaminen peruskoulussa on tärkeää?”* ja *”mitä vaikutuksia ohjelmointiosaamisella on oppilaiden tulevaisuudelle?”*. Kummallakin kysymyksellä pyrittiin löytämään vastauksia samaan tutkimuskysymykseen, joten kysymyksiin saadut vastaukset yhdistettiin aineiston analyysissä. Saadut vastaukset jakautuivat päämääriin *oppilaan-*, *yhteiskunnan-* ja *koulun* kannalta, painottuen kuitenkin päämääriin oppilaan kannalta. Ohjelmoinnin opetuksen päämääriksi oppilaan kannalta muodostettiin vielä vastausten *(n=42)* perusteella yläkäsitteet, jotka olivat *monilukutaito (n=33), ajattelun taidot (n=31), työ- ja elinkeinoelämä (n=27), elämänhallintataidot (n=12)* sekä *ohjelmointitaito (n=11)*. Vähäisiä mainintoja *(n=1–8)* ohjelmoinnin opetuksen päämääriksi saivat myös *tulevaisuuden taidot*, *luovuus*, *sosiaalisuus*, *turvataidot* sekä *matemaattiset taidot*.

6.3.1 Ohjelmoinnin opetuksen päämääränä monilukutaidon kehittyminen

Eniten mainintoja *(n=33)* ohjelmoinnin opetuksen päämääristä kohdistui monilukutaidon kehittymiseen. Ohjelmoinnin oppimista kuvattiin tulevaisuuden kansalaistaitona, jossa ymmärretään ympäröivän informaatioyhteiskunnan toimintaperiaatteita ja sitä, miten ohjelmoidut ratkaisut vaikuttavat yksittäisen ihmisen ja sitä kautta koko yhteisön arkeen. Vastaajat 26 ja 19 kuvasivat monilukutaidon ja ohjelmoinnin opetuksen yhteyksiä esimerkiksi seuraavin tavoin:

> *”Koska maailmamme rakentuu enemmän ja enemmän tietotekniikan varaan, on jokaisen tärkeä ymmärtää millä perusperiaatteilla ohjelmat toimivat, samaan tapaan kuin yleissivistykseen kuuluu ymmärtää perusteet biologiasta, fysiikasta ja kemiasta” (Vastaaja 26, aineenopettaja)*

”Digiaikana tieto on valtaa - ohjelmoitujen ratkaisujen vaikutuksen ymmärtäminen yksittäisen ihmisen ja yhteisöjen arkeen liittyen. Koska tietokoneet, verkkoyhteydet ja tekoäly ovat niin suuri tekijä nyt ja tulevaisuudessa, oppilaiden tulee ymmärtää miten ne toimivat myös kooditasolla.” (Vastaaja 19, pedagoginen asiantuntija)

Vastaajat mainitsivat ohjelmoinnin päämääriksi monilukutaitoon liittyen teknologisen ymmärryksen, teknologisen yleissivistyksen ja teknologisen lukutaidon kehittymisen. Nämä sisältävät arkielämän elektronisten laitteiden, tietokoneiden ja automaation ohjaamisen ymmärtämisen ohjelmoinnin kooditasolla. Toisaalta monilukutaitoon liitettiin myös ohjelmointiin liittyvien rajoitusten ymmärtämisen. Vastauksissa mainittiin myös ohjelmoinnin oppimisen hälventävän teknofobiaa, kun ymmärrys ohjelmoidun maailman toimintaperiaatteista selkiytyy.

”Ehkä nykyaikaisen ihmisen on tärkeää ymmärtää ohjelmoinnillista ajattelua yhteiskunnan automatisoitumisen ja esimerkiksi tekoälyn kehittymisen vuoksi” (Vastaaja 39, luokanopettaja)

Ohjelmoinnin nähtiin toisaalta liittyvän nyky-yhteiskunnassa kaikkeen, joten sen opettamisen koettiin tukevan uuden oppimista esimerkiksi ohjelmien käytön suhteen. Ohjelmoinnin oppimisen päämäärinä mainittiin myös pelien mekaniikkojen ymmärtäminen ja uusmedian lukutaidon kehittyminen.

”Ohjelmointiajattelu, jotta ymmärtää mitä se on ja perusteet miten ohjelmoidaan. Ohjelmointi ja sen ymmärtäminen liittyy nyky-yhteiskunnassa kaikkeen.” (Vastaaja 18, aineenopettaja)

6.3.2 Ajattelun taitojen kehitys ohjelmoinnin opetuksen päämääränä

Ohjelmoinnin oppimisen päämääriksi mainittiin usean vastaajan *(n=31)* toimesta ajattelun taitojen kehittyminen. Vastaajat kuvasivat ohjelmoinnin oppimisen kehittävän itsenäistä ajattelua kokonaisvaltaisesti. Sillä nähtiin olevan vaikutuksia ohjelmoinnillisen, algoritmisen ja loogisen ajattelun sekä ongelmanratkaisutaitojen kehittymisen kannalta. Ajattelun taitojen kehitystä ja ohjelmoinnin oppimisen yhteyttä kuvattiin muun muassa tällä tavoin:

”Ymmärrys ohjelmien ja digitalisaation vaikutuksesta yhteiskuntaan, ymmärrys ohjelmien ja tietokoneiden toiminnasta ja toisaalta toiminnan rajoista, tietokoneen käyttötaidot, kyky kirjoittaa ohjeita auki” (Vastaaja 22, aineenopettaja)

”Oppilaat oppivat pilkkomaan ongelmia osiin ja keksimään ratkaisuja.” (Vastaaja 4, luokanopettaja)

”Joku voi löytää intohimonsa, toinen voi kehittyä ongelmanratkaisussa, jota tarvitaan kaikilla elämän osa-alueilla.” (Vastaaja 41, aineenopettaja)

Ohjelmoinnin oppimisen kerrottiin myös kehittävän kykyä kirjoittaa ohjeita auki, pilkkoa ongelmia auki ja huomata syy-seuraussuhteita. Ohjelmoinnin oppiminen kehittää vastaajien mukaan myös kykyä soveltaa ajattelun taitoja yksilön arjessa ongelmanratkaisutilanteissa. Vastaajat näkivät ohjelmoinnin oppimisen myötä kehittyneiden ajattelun taitojen olevan tärkeässä osassa myös tulevaisuuden työelämän kannalta.

6.3.3 Työ- ja elinkeinoelämän vaikutukset ohjelmoinnin opettamisen päämääriin

Ohjelmoinnin oppimisella nähtiin olevan vaikutuksia oppilaan tulevaisuudelle kouluttautumisen ja työllistymisen suhteen. Ohjelmoinnin opetuksen nähtiin vaikuttavan oppilaiden mielenkiinnon ja innostuneisuuden kasvamiseen ohjelmointiin liittyviä aloja kohtaan sekä parantavan heidän mahdollisuuksiaan työ- ja koulutusmarkkinoilla. Vastaajien mukaan ohjelmoinnin osaamisen nähtiin madaltavan kynnystä hakeutua jatkokoulutukseen. Kouluttautuminen ohjelmointialalle nähtiin myös liittyvän parempiin työllistymisnäkymiin.

”Att ge dem bättre chans på arbetsmarknaden.” (Vastaaja 38, aineenopettaja)

”Paljon hyviä vaikutuksia. Työllistyminen, laaja-alaisempi motivaatio jatko-opintoihin. Ohjelmointia tarvitaan useilla aloilla.” (Vastaaja 24, luokanopettaja)

”Hakeutuminen itseä kiinnostavalla alalle helpottuu kun on pienet pohjatiedot ohjelmoinnistakin.” (Vastaaja 7, aineenopettaja)

”Madaltaa rimaa hakeutua eri alojen jatko-opintoihin.” (Vastaaja 20, aineenopettaja)

Vastaajien mukaan työtehtävät nyt ja tulevaisuudessa sisältävät tai vaativat ohjelmointiosaamista ja kykyä ohjelmoinnilliseen ajatteluun. Ohjelmointiosaamisen nähtiin vastaajien mukaan olevan yhteydessä tulevaisuuden työelämän tarpeisiin, kun algoritmien ja ohjelmistojen ymmärtäminen tulee yhä tärkeämmäksi.

> *”Tehokkuuden lisääntyessä mekaanisessa työnteossa ohjelmoinnin tai ainakin ohjelmallisen ajattelun taidot tulevat olemaan tärkeitä monilla aloilla” (Vastaaja 17, aineenopettaja)*

> *”Melko monenlaiset työtehtävät saattavat tulevaisuudessa sisältää ohjelmointiin liittyviä asioita.” (Vastaaja 39, luokanopettaja)*

Osa vastaajista näki ohjelmoinnin opettamisen innostavan oppilaita ohjelmoimaan myös vapaa-ajalla harrastuksen omaisesti:

> *”Yksi potentiaalinen vaihtoehto uravalinnaksi tai harrastukseksi.” (Vastaaja 13, luokanopettaja)*

> *”Siitä innostuneet saavat vapaa-ajan harrastuksen ja kenties ammatin.” (Vastaaja 31, aineenopettaja)*

6.3.4 Ohjelmoinnin opettaminen elämänhallinnan taitojen kehittäjänä

Vastaajien mukaan ohjelmoinnin oppimisella nähtiin olevan vaikutuksia oppilaan elämänhallintataidoille. Vastaajat kertoivat ohjelmoinnin opetuksen vahvistavan oppilaan opiskelun perustaitoja, kuten tavoitteen asettamista, virheistä oppimista, ohjeiden noudattamista, keskittymiskykyä, huolellisuutta, suunnitelmallisuutta ja ajattelun joustavuutta.

> *”Ongelmanratkaisukyvyn ja siihen liittyvän sinnikkyyden kehittäminen. Parhaimmillaan oppilas oivaltaa sinnikkyyden ja työn tekemisen merkityksen kehityksen ja menestymisen kannalta.” (Vastaaja 43, aineenopettaja)*

Lisäksi sen nähtiin kehittävän oppilaan pitkäjänteisyyttä, sinnikkyyttä ja pettymysten sietokykyä. Vastaajat mainitsivat myös ohjelmoinnin oppimisen lisäävän oppilaan itseluottamusta, itsetuntoa sekä uskoa omiin kykyihin.

”Lisää itseluottamusta omiin mahdollisuuksiin ja kykyihin. Vahvistaa opiskelun perustaitoja kuten tavoitteen asettaminen, keskittyminen, suunnitteleminen, ajattelun joustavoituminen.” (Vastaaja 14, aineenopettaja)

6.3.5 Ohjelmointitaito kehittyy ohjelmoimalla

Vastaajat näkivät ohjelmoinnin opetuksen päämääräksi myös itse ohjelmointitaidon kehittymisen. Ohjelmointitaitoa pidetiin osana yleissivistävää teknistä kasvatusta, joka pitää sisällään ohjelmoinnin perusteita. Vastaajien mukaan ohjelmointitaito auttaa myös ymmärtämään ohjelmoinnin mahdollisuuksia ja sen avulla toteutettavia prosesseja. Peruskoulun ohjelmoinnin opetuksen tehtävänä nähtiin tutustuttaa oppilaat ohjelmoinnin alkeisiin. Vastaajat näkivät ohjelmoinnin opetuksen päämääriksi myös sosiaalisten ja matemaattisten taitojen sekä luovuuden kehityksen. Myös turvataidot ja tietokoneen käyttötaidot mainittiin osana ohjelmoinnin opetuksen päämääriä.

”Peruskoulussa pitäisi avata ohjelmointia niin, että jokainen ymmärtää, että kyseessä on opittavissa oleva taito, eikä mikään mahdottoman vaikea salatiede. Tavoite on tehdä ylläoleva motivoivalla tavalla ja nimenomaan niin, että opitaan tekemällä. Pelkkä teoria ei riitä tässä tapauksessa.” (Vastaaja 26, aineenopettaja)

”Alaluokilla se on lähinnä tutustumista ja kiinnostuksen herättämistä ohjelmointia kohtaan. On tärkeää, että oppilaan kiinnostus ja käsitys itsestä "ohjelmoijana" kehittyy.” (Vastaaja 25, luokanopettaja)

”Tehokkuuden lisääntyessä mekaanisessa työnteossa ohjelmoinnin tai ainakin ohjelmallisen ajattelun taidot tulevat olemaan tärkeitä monilla aloilla, myös esim taito- ja taidealoilla. Siksi tarvitaan yhteiset pelisäännöt perusopetukseen.” (Vastaaja 17, aineenopettaja)

6.3.6 Muita ohjelmoinnin opettamisen päämääriä

Muutama vastaajista mainitsi ohjelmoinnin opetuksella olevan päämääriä myös yhteiskunnallisella ja koulun sisäisellä tasolla. Ohjelmointiosaamisen nähtiin tuovan tasa-arvoa ja vapautta toimia itsenäisesti teknologisoituvassa yhteiskunnassa:

”Kyseessä on vapaus ja demokratia: Kun ymmärrämme nämä prosessit ja rakenteet, niiden käyttäminen meitä vastaan on haastavampaa.” (Vastaaja 19, pedagoginen asiantuntija)

Ohjelmointiosaamisen nähtiin auttavan vallan jakautumisessa tasaisesti ja mahdollistavan demokratian. Se nähtiin osallisuutta ja voimaantumista lisäävänä tekijänä yhteiskunnallisella tasolla. Yksi vastaajista nosti esiin myös yhteiskunnallisen pulan osaavista ohjelmoijista:

”Algoritmien ymmärrystä ja ohjelmointia tarvitaan nykyään monella alalla ja jatkossa todennäköisesti vielä enemmän. Osaavista ohjelmoijista on myös pulaa.” (Vastaaja 26, aineenopettaja)

Yksi vastaajista mainitsi ohjelmoinnin opetuksen päämääräksi yhdistää oppiaineita monialaisiksi kokonaisuuksiksi:

”Monialaisuus ja ohjelmointi - ohjelmointi oppiaineita yhdistävänä työkaluna.” (Vastaaja 19, pedagoginen asiantuntija)

7. Päätelmät

Tässä luvussa esitellään tutkielman johtopäätökset, niiden pohdinta sekä arvioidaan tutkielman luotettavuutta ja eettisiä ratkaisuja. Ensimmäisessä alaluvussa vastataan tutkimuskysymyksiin johtopäätöksien ja pohdinnan muodossa pohtien tulosten syitä ja seurauksia sekä suhteutetaan niitä aikaisempaan tutkimustietoon. Toinen alaluku käsittelee tutkielman luotettavuutta, toistettavuutta ja jatkotutkimusmahdollisuuksia.

7.1 Tutkimuskysymyksiin vastaaminen

Tutkielman tavoitteena oli selvittää, millaisin eri tavoin ohjelmointia opetetaan suomalaisissa peruskouluissa. Tutkielmassa tarkasteltiin suomalaisissa peruskouluissa tapahtuvassa ohjelmoinnin opetuksessa esiintyviä käsitteitä, käytänteitä ja päämääriä. Vastauksia etsittiin tutkimuskysymysten ”Mitkä ohjelmointiin liittyvät käsitteet korostuvat ohjelmoinnin opetuksessa?”, ”Millaisten tehtävien ja työtapojen avulla ohjelmointia opetetaan?” ja ”Millaisia näkemyksiä opettajilla on ohjelmoinnin opetuksen keskeisistä tavoitteista ja vaikutuksista oppilaiden tulevaisuudelle?” avulla.

7.1.1 Ohjelmoinnin opetuksen käsitteet

Kolme eniten mainittua käsitettä olivat ohjelmointikielissä ja -ympäristöissä esiintyvät termit ehto, toisto ja muuttuja. Vastaajat nimesivät tärkeimmiksi käsitteiksi suoraan tekemiseen, eli ohjelmointiin liittyviä käsitteitä. Tulosten perusteella ohjelmoinnin opetuksessa käytettävät käsitteet liittyvät siis suoraan substanssiin eli itse ohjelmointiin. Seuraavaksi tärkeimpinä käsitteinä tulosten perusteella olivat algoritmin ja komennon käsitteet. Yhtä tärkeinä ei pidetty ohjelmoinnin opetuksen päämäärinäkin esiintyviä ajattelun taitoja ja niiden käsitteitä. Vaikka tulosten perusteella vastaajat korostivatkin ohjelmoinnin opetuksen päämäärinä itse ohjelmointitaidon ulkopuolisia tekijöitä, kuten esimerkiksi ajattelun taitoja, eivät nämä päämääriin liittyvät käsitteet korostuneet erityisesti ohjelmoinnin opetuksessa käytettävien käsitteiden saralla.

Ajattelun taitoihin liittyviä käsitteitä mainittiin yhdeksässä ohjelmoinnin opetuksen käsitteitä käsittelevässä vastauksessa, ja ne luokiteltiin kaikki saman kattokäsitteen, ajattelun taidot, alle.

Esimerkiksi käsite *algoritminen ajattelu* luettiin osaksi tätä kattokäsitettä. Kuitenkin algoritminen ajattelu mainittiin näistä yhdeksästä vastauksesta vain kahdessa. Ajattelun taitoja kuvaavat yhdeksän vastausta oli siis koottu yhteen samaan klusteriin niiden aihepiirin perusteella.

Tämän tutkielman tuloksissa korostuneet käsitteet tukevat aiemmissa tutkimuksissa (ks. Brennan & Resnick, 2012) ja julkaisuissa (ks. Mykkänen & Liukas, 2014) mainittuja keskeisiä ohjelmoinnin käsitteitä. Nämä käsitteet eivät kuitenkaan selkeästi löydy esimerkiksi Perusopetuksen opetussuunnitelman perusteista (OPH, 2014) eivätkä Uudet lukutaidot -kehittämisohjelman ohjelmointiosaamisen kuvauksista. Näiden käsitteiden mainitseminen osana ohjelmoinnin opetuksen sisältöjä tai ohjelmointiosaamisen voisi yhtenäistää ohjelmoinnin opetuksen sisältöjä. Vastaavana esimerkkinä matematiikan geometrian ja mittaamisen sisällöissä Perusopetuksen opetussuunnitelman perusteissa mainitaan kirjaimellisesti käsitteet piste, suora ja jana (OPH, 2014, s. 236).

Vaikka vastaajat eivät korostaneetkaan vastauksissaan ohjelmointiopetuksen taustalla olevia käsitteitä koskien ohjelmoinnin opetuksen työtapoja tai päämääriä, olivat ne silti vastaajilla muiden kysymysten vastausten perusteella selkeästi hallinnassa. Kyselytutkimuksen heikkoutena tämän ohjelmoinnin opetuksen käsitteiden tutkimisen kohdalla oli se, että vastaajat kertoivat viisi tärkeimpänä pitämäänsä käsitettä omien ajatustensa pohjalta. Näin ollen vastauksista ei välttämättä selviä juuri ne käsitteet, jotka todellisuudessa korostuvat osana ohjelmoinnin opetusta. Esimerkiksi havainnoimalla oikeita ohjelmointiopetuksen tilanteita, olisi mahdollisesti saatu laajempi tai eriävä aineisto ohjelmoinnin opetuksessa korostuvista ohjelmoinnin käsitteistä.

7.1.2 Ohjelmoinnin opetuksen käytänteet

Tulosten perusteella ohjelmoinnin opetuksen käytänteet jakautuvat useaan eri kategoriaan. Vastauksissa käytänteitä saatiin opettajan toimintaan, työskentelymuotoihin, tehtävätyyppeihin sekä oppimisen strategioihin liittyen. Tulosten mukaan ohjelmoinnin opetuksen käytänteisiin sisältyy siis paljon erilaisia ja toisistaan irrallisia lähtökohtia ja näkemyksiä. Vastausten laaja kirjo antoi myös näkemyksen siitä, että erilaisia ohjelmoinnin opetuksen käytänteitä on olemassa yhtä paljon kuin on erilaisia opettajiakin. Vastauksista oli luettavissa vastaajien opettavan ohjelmointia hyödyntäen omia pedagogisia näkemyksiään.

Vastaajat lähestyivät ohjelmointitehtäviä antaen esimerkkejä, ongelmalähtöisiä tehtävänantoja, käyttäen valmista materiaalia, tai lähestyen tehtäviä yhteisesti avaten ohjelmien taustalla olevaa logiikkaa. Osa lähestyi ohjelmointia suunnitelmallisesti tarkastellen koodia esimerkiksi pseudokoodin kirjoittamisen tai vuokaavioiden avulla. Tuloksissa vastaajat kertoivat myös liittäneensä ohjelmointitehtäviä osaksi oppilaiden arkielämää. Arkielämä ja ohjelmointi liitettiin toisiinsa esimerkiksi yksityiskohtaisten toimintaohjeiden, kuten aamutoimien tai leivontaohjeiden, avulla. Ohjelmointia lähestyttiin myös valmiin koodin tutkimisen avulla. Aineistosta saatujen tulosten mukaan ohjelmoinnin opetuksessa käytetään sekä yksilö- että pari- ja ryhmätyöskentelymuotoja. Valittuja työskentelymuotoja perusteltiin esimerkiksi eriyttämisen tai yhteistoiminnallisen oppimisen näkökulmista.

Ohjelmointia opetetaan laajasti käyttäen erilaisia tehtävätyyppejä hyödyntäen eri sovelluksia ja laitteita, sekä koneetonta ohjelmointia. Tulosten mukaan ohjelmoinnin opetuksessa hyödynnetään usein projektinomaisia tehtävätyyppejä, joita ovat esimerkiksi animaatioiden, pelien ja erilaisten sovelluksien ohjelmoiminen. Myös yksittäisiä tehtävänantoja ja lyhyempiä ohjelmointitehtäviä kuvattiin vastauksissa. Tulosten perusteella ohjelmoinnin opetuksessa hyödynnetään niin itsetehtyjä ja suunniteltuja tehtävänantoja kuin valmiita opetusmateriaalejakin, sekä oppilaiden omia projekteja, jotka perustuvat vapaampiin tehtävänantoihin. Kaikkia näitä käytettyjä tehtävätyyppejä kuvattiin perustellen.

Ohjelmointia lähestyttiin myös huomioiden erilaisia ohjelmointioppimisen strategioita. Vastaajat ohjasivat tietoisesti oppilaat erilaisten tehtävänantojen avulla käyttämään erilaisia ohjelmoinnin oppimisen strategioita, jotka ovat löydettävissä myös ohjelmoinnin oppimisen teoriasta (ks. 3.3. Ohjelmointi ja ohjelmoinnillisen ajattelun yhteys). Tulosten mukaan ohjelmoinnin opetuksessa hyödynnetään *testaamista ja virheenkorjausta*, *muokkaamista ja uudelleenkäyttämistä* sekä *kokeilemista ja iterointia* (ks. Brennan & Resnick, 2012). Näitä ohjelmointioppimisen strategioita hyödynnettiin useissa eri tehtävätyypeissä.

Tutkielman tulokset ohjelmoinnin käytänteiden osalta tukevat aikaisempien tutkimusten tuloksia. Esimerkiksi ohjelmoinnin opetukselle toimivat strategiat tai opetusmallit kuten koneeton ohjelmointi, yhteistoiminnallinen oppiminen, ohjelmoinnilliseen ajatteluun opettaminen, ohjelmointitehtävien liittäminen osaksi oppilaan arkielämää sekä valmiin koodin tutkiminen (Sentace & Csizmadia, 2016) löytyvät myös tämän tutkielman tuloksista. Aiemmissa tutkimuksissa

esitetyt toimintamallit ohjelmoinnin oppimiseksi, kuten itsenäinen koodaaminen visuaalisissa ohjelmointiympäristöissä sekä robotteja rakentamalla ja ohjelmoimalla (Wu ym. 2019), toteutuvat tutkielman tulosten mukaan suomalaisissa peruskouluissa. Tutkielman tuloksien mukaan yhteistoiminnallista oppimista hyödynnetään ohjelmoinnin opettamisessa, mikä tukee myös aikeisempaa tutkimusta (Wu ym., 2019; Otterborn ym., 2019).

Aiempien tutkimusten mukaan suomalaiset opettajat suosivat osallistavia tehtävätyyppejä, joissa oppilaat pääsevät itse ohjelmoimaan kirjoittaen koodia tai rakentaen ja ohjelmoiden robotteja (Wu ym., 2019). Tämän tutkielman tulokset vahvistavat aiempaa teoriaa aiheesta, tutkielman tulosten antaessa vastaavan kuvan ohjelmoinnin opetuksessa käytettävistä tehtävätyypeistä. Myös tutkielman tuloksista löytyvät ohjelmoinnin oppimisen strategiat tukevat ja toisaalta tukeutuvat aiempaan teoriaan (ks. Sentace & Csizmadia, 2016; Brennan & Resnick, 2012). Vastaajat kertoivat monipuolisesti perustellen käyttämiään strategioita, esimerkiksi virheenkorjaamista, ja toisaalta kertoivat näiden oppimisen strategioiden toimivan myös luonnollisina oppimisen mekanismeina osana ohjelmoinnin oppimista. Jälleen havainnoimalla oikeita ohjelmointiopetuksen tilanteita tai teemahaastatteluiden avulla, olisi mahdollisesti saatu tarkempia kuvauksia erilaisista ohjelmointiopetuksen käytänteistä. Kyselytutkimuksella saatiin kuitenkin tutkielman aikatauluun nähden kattava ja laaja aineisto erilaisista ohjelmoinnin opetuksen käytänteistä.

7.1.3 Ohjelmoinnin opetuksen päämäärät

Ohjelmoinnin opetuksella nähtiin olevan useita eri päämääriä ja tavoitteita. Päämäärinä vastauksissa painottuivat monilukutaidon ja ajattelun taitojen kehitys sekä työ- ja elinkeinoelämään liittyvät näkökulmat. Vastaajat näkivät ohjelmoinnin opetuksen päämäärät siis itse ohjelmointitaidon ulkopuolisina, tulevaisuuteen ja yksilön kehitykseen suuntaavina päämäärinä. Tulosten perusteella vastaajat näkivät ohjelmoinnin opetuksen tärkeänä tekijänä tulevaisuuden yhteiskuntaan kasvattamisessa, sen kehittäessä esimerkiksi ohjelmoinnillista ajattelua. Vastaajat näkivät ohjelmoinnin opettamisen kantavan hedelmää oppilaiden tulevaisuudessa.

Monilukutaidon kehityksen kannalta vastaajat näkivät merkitykselliseksi yhteiskuntatoimijuuden, teknologisen lukutaidon ja ympäröivän informaatioyhteiskunnan toimintaperiaatteiden

ymmärryksen. Monilukutaidon kehittymisen liitettiin myös ymmärrys teknologian vaikutuksista yksilöön ja yhteisöön. Tuloksissa ajattelun taitoihin liitettiin kyky itsenäiseen ajatteluun, ongelmaratkaisutaitoon, loogiseen ajatteluun, ohjelmoinnilliseen ja algoritmiseen ajatteluun sekä näiden ajattelun taitojen soveltamiseen. Ohjelmoinnin opetuksen työ- ja elinkeinoelämään liittyvät päämäärät liittyivät helpompaan työllistymiseen, madaltuneeseen kynnykseen hakeutua jatko-opintoihin sekä mielenkiinnon heräämiseen ohjelmointialaa kohtaan. Myös työelämän ja koulutuksen tarpeiden muutoksen kohtaaminen nähtiin ohjelmoinnin opetuksen päämääränä. Tulokset tukevat teoriaa, jossa ohjelmointitaitojen ja ohjelmoinnillisen ajattelun taidot nähdään osana tulevaisuuden taitoja (ks. ISTE, 2021; McClelland & Grata, 2018; Bocconi ym., 2016; Mohaghegh & McCauley 2016; McCormack, 2014) ja niiden nähtiin myös kytkeytyvän yksilön kykyyn toimia, osallistua ja innovoida digitalisoituvassa yhteiskunnassa (ks. Bocconi ym., 2016, The White House, 2016.)

Ohjelmoinnin opetuksen nähtiin kehittävän myös elämänhallintataitoja. Vastaajat kertoivat ohjelmoinnin opetuksen kehittävän oppilaiden opiskelun perustaitoja, kuten tavoitteiden asettamista, virheistä oppimista, sinnikkyyttä, keskittymiskykyä, suunnitelmallisuutta ja ajatusten joustavuuden lisääntymistä. Ohjelmoinnin opetuksen katsottiin myös kehittävän oppilaiden itseluottamusta.

Edellä mainittujen päämäärien lisäksi ohjelmoinnin opetukselle koettiin tavoitteeksi myös itse ohjelmointitaidon lisääminen. Ohjelmointitaito liitettiin yleissivistävään tekniseen kasvatukseen, jolla pyritään kehittämään oppilaiden ymmärrystä ohjelmoinnin mahdollisuuksista ja sen avulla toteutettavista prosesseista. Ohjelmoinnin opetuksen peruskoulukontekstissa nähtiin toimivan tutustuttajana ohjelmoinnin alkeisiin, joka toimii pohjana kumulatiiviselle ohjelmoinnin oppimiselle oppilaiden tulevaisuudessa.

Tutkielman tulokset ohjelmoinnin opettamisen päämäärien osalta vastaavat aiempaa teoriaa ohjelmoinnin opettamisen päämääristä esimerkiksi digitaaliseen yhteiskuntaan kasvattamisen, ohjelmointiin liittyvien taitojen ja tavoitteiden sekä ajattelun taitojen suhteen (ks. Otterborn, ym., 2019). Myös Brennanin ja Resnickin (2012) tutkimuksessa esiintyneet ohjelmoinnillisen ajattelun kehittymisen näkökulmat itseilmaisun ja kyseenalaistaminen suhteen olivat nähtävissä tutkielman tuloksissa. Tulosten mukaan vastaajat opettavat ohjelmointia pääsääntöisesti oppilaiden monilukutaidon ja ajattelun taitojen kehittymisen kannalta sekä työ- ja elinkeinoelämän

tarpeita ja vaateita huomioiden, itse ohjelmointitaidon jäädessä hieman sivuun ohjelmoinnin opetuksen tavoitteiden keskiöstä.

7.2 Tutkimuksen yhteenveto

Tämän tutkielman tavoitteena oli selvittää, millaisin eri tavoin ohjelmointia opetetaan suomalaisissa peruskouluissa. Tutkielman teon taustalla oli ymmärrys siitä, että Perusopetuksen opetussuunnitelman perusteet (OPH, 2014) tarjoamat tavoitteet ja sisällöt eivät ole vielä tarjonneet riittäviä lähtökohtia yhtenäiselle ohjelmointiopetukselle maanlaajuisesti (OPH, 2021). Myös Digiajan peruskoulu II -raportin mukaan ohjelmointiin liittyvien laaja-alaisten osaamistavoitteiden toteuttamiseksi tulisi valmistella valtakunnallisesti yhteneväinen ohjeistus (Tanhua-Piiroinen ym., 2020).

Kaikkea tutkimusta tehdessä pyritään välttämään virheitä, mistä syystä yksittäistä tutkimusta tehdessä on arvioitava sen luotettavuutta. Vaikka laadullisen tutkimuksen luotettavuuteen ei ole olemassa yksiselitteisiä ohjeita, on luotettavuuden arvioinnissa syytä huomioida ainakin tutkimuksen tarkoitusta, tutkijoiden sitoutumista, aineiston keruuta ja aineiston analyysiä. (Tuomi & Sarajärvi, 2018). Tutkimuksen validiteetilla ja reliabiliteetilla tarkoitetaan tutkimuksen pätevyyttä ja toistettavuutta. Validiteetiin liittyy teoreettisen ja käsitteellisen määrittelyn yhtenäisyys tehtyjen tulkintojen ja johtopäätösten suhteen. Reliabiliteetilla tarkoitetaan tutkimuksen toistettavuuden arvioimista huomioimalla esimerkiksi käytettyjä mittareita tai käytettyä havainnointimenetelmää. Tutkielman luotettavuutta voidaan tarkastella tutkimuksen vahvuuksien perustelemisella, mutta toisaalta myös tutkimuksen rajoituksia arvioimalla. (Eskola & Suoranta, 2014). Tutkielman luotettavuutta arvioidaan tutkijoihin, tutkimuksen tekoon, vastaajiin sekä kyselylomakkeeseen liittyvien seikkojen kautta.

Tutkielman tuloksia lukiessa on hyvä pitää mielessä, ettei niitä voida yleistää koskemaan kaikkia suomalaisia peruskouluoja, sillä tutkielman aineisto kerättiin lähteistä, joiden jäsenet ovat todennäköisemmin tavallista kiinnostuneempia ohjelmoinnin opetuksesta. Tutkielman tuloksista voidaan kuitenkin nähdä suhteellisen laajasti ohjelmoinnin opetuksen sisältöjen ja toimintatapojen variaatioita peruskoulukontekstissa. Tutkielmamme tuloksia voidaan hyödyntää esimerkiksi ohjelmointiopetuksen oppimateriaalin tuotannossa pohtiessa, minkälaiset tehtävätyypit on jo koettu hyväksi peruskoulun ohjelmoinnin opetuksen kentällä. Tutkielman tulokset

vahvistavat myös mielikuvaa opettajien tulevaisuuteen suuntaavista pedagogisista näkemyksistä myös ohjelmoinnin opetuksen saralla. Lisäksi ohjelmoinnin opettamisesta kiinnostunut, mutta ei vielä harjaantunut, opettaja voi löytää tutkielmasta uusia ajatussuuntia tai ideoita omaan työhönsä ja ohjelmoinnin opettamisen järjestämiseen.

Tutkielmassa tapausta tutki koko tutkimusprosessin ajan yhtäaikaisesti kaksi tutkijaa, jotka muodostivat tutkijatriangulaation. Esimerkiksi Eskola ja Suoranta (2014) toteavat teoksessaan kahden havainnoitsijan tai menetelmän käytön voivan parantaa tutkimuskohteen kuvausta ja tutkimuksen objektiivisuutta. Tutkijat lähestyivät tutkimuskohteena ollutta tapausta, ohjelmoinnin opetusta suomalaisissa peruskouluissa, laajasti sekä kansainvälisiä että kansallisia lähteitä hyödyntäen. Tutkijoiden omat intressit aihepiiriä kohtaan olivat myös tutkimuksen kannalta kantavana voimavarana niiden toimiessa sitouttavina tekijöinä. Toisaalta ne saattoivat aiheuttaa myös subjektiivisten näkemysten alitajuntaisen vaikutuksen tutkimuksen tuloksiin itse analyysivaiheessa, vaikka tutkijat pyrkivätkin jatkuvaan objektiivisuuteen tutkielman teossa. Tutkijatriangulaation olemassaolo kuitenkin vähensi tämän subjektiivisuuden vaaraa sen mahdollistaessa ajatustenvaihdon. (Tuomi & Sarajärvi, 2018.) Kirjoitusprosessin aikana tutkijat kirjoittivat tutkielmaa sekä yhdessä että erikseen, tekstejä keskenään vaihdellen, muokaten ja kommentoiden.

Laadullisessa tutkimuksessa analyysin ja menetelmien luotettavuuteen vaikuttaa niiden tekeminen läpinäkyviksi, sekä tutkijan ymmärrys itsestään subjektiivisena toimijana vastausten tulkitsijana (Hirsjärvi ym., 2010). Koska laadullisen tutkimuksen alueella on löydettävissä vain vähän standardoituja tapoja analysoida aineistoa (Puusa & Juuti, 2020), on tulosten raportoinnin luotettavuutta pyritty lisäämään tekemään siitä mahdollisimman avointa ja yksityiskohtaista (ks. Tuomi & Sarajärvi, 2018). Aineiston analyysissa käytetty teoriaohjaava sisällönanalyysi perustuu osittain tutkijan subjektiiviseen näkemykseen aineiston sisällöstä, tutkijan joutuessa itse päättämään ja erottelemaan aineistosta sen tärkeimpänä pitämänsä sisällön (Tuomi & Sarajärvi, 2018). Tutkijat olivat kuitenkin perehtyneet jo etukäteen tutkimansa aihepiirin teoriaan, joka lisää tutkielman tulosten luotettavuutta.

Tutkimuksessa tutkittiin ohjelmoinnin opetusta suomalaisen peruskoulun kontekstissa, ja haluttiin selvittää, millaisin eri tavoin ohjelmointia opetetaan suomalaisissa peruskouluissa. Tut-

kimusjoukkona toimineet ohjelmointia peruskouluissa opettaneet vastaajat tukivat siis tutkimuksen tarkoitusta. (Tuomi & Sarajärvi, 2018). Tutkielman aineisto kerättiin vuoden 2021 maalis-huhtikuun aikana, jolloin ohjelmointi oli kuulunut osaksi perusopetusta jo lähes viiden vuoden ajan. Näin pitkän ajanjakson aikana ohjelmoinnin opetus on jo ehtinyt saada jalansijaa ja muotoutua perusopetuksen parissa, joten tutkimuksen tekeminen ohjelmoinnin opetuksen eri tavoista suomalaisissa peruskouluissa oli mielekästä. Tutkielman aineiston koko *(n=42)* oli kvalitatiiviselle tutkimukselle ominaisesti pienehkö (Eskola & Suoranta, 2014) mutta kuitenkin kohtuullinen. Tutkimuksen tuloksia ei voida otantakoon vuoksi nähdä yleistettävänä koko maata kattaviksi tuloksiksi, mutta se ei toisaalta ollut myöskään tutkimuksen tarkoitus. Kvalitatiivisen tutkimuksen tavoin tutkielman tarkoitus oli tarkastella tapausta, ohjelmoinnin opetusta, luomatta maanlaajuisia yleistyksiä sen nykytilasta. Kuitenkin otannan laajuus ja sen kertyminen eri puolilta Suomea antaa todennäköisesti hyvin kuvan siitä, millaisin eri tavoin ohjelmointia opetetaan suomalaisissa peruskouluissa.

Tutkimukseen vastattiin anonyymisti nettilinkin kautta. Mahdollisuus vastata kyselyyn anonyymisti tekee kyseenalaiseksi sen, ovatko annetut vastaukset aitoja mielipiteitä ja asianmukaisia vastauksia. Kyselyn osallistumislinkki jaettiin kuitenkin sosiaalista mediaa hyödyntäen ryhmittymiin, joiden jäsenillä oli todennäköisesti jonkinlaista taustaa tai harrastuneisuutta sekä positiivinen asenne ohjelmoinnin opettamista kohtaan. Tämän ennakkoasetelman voidaan nähdä lisätä tutkimuksen luotettavuutta lisäävänä tekijänä, vastaajien todennäköisen innostuneisuuden ja aihepiiriin perehtyneisyyden kautta. Lisäksi vastaajien vastausinnokkuuteen pyrittiin vaikuttamaan liittämällä kyselylinkin yhteyteen erilaisia motivoivia kuvia (ks. liite 2), jonka voitiin nähdä lisäävän vastaajien määrää sosiaalisen median alustoilla saatujen kommenttien perusteella.

Koska yli puolet vastaajista *(n=24)* toimi aineenopettajan yläkoulussa, ja luokanopettajien osuus oli 17 vastaajaa, on todennäköistä, että tulokset heijastelevat hieman enemmän yläkoulussa tapahtuvaa ohjelmoinnin opetusta. Koska vastaajia oli kuitenkin suhteellisen paljon sekä ylä- että alakoulun puolelta, voidaan aineistoa pitää kuitenkin kattavana ja antavan laajan kuvan ohjelmoinnin opetuksen eri variaatioista alaluokilta yläluokille asti.

Tutkimuksen aineisto kerättiin sähköisenä kyselynä Webropol -kyselytyökalun avulla. Kyselyn alussa vastaajille kerrottiin tutkimuksen tarkoitus, tietojenkäsittelymenetelmät ja ilmoitettiin,

että vastaamalla kyselyyn vastaaja antaa luvan aineiston käsittelyyn kyseisessä tutkielmassa. Kyselystä saatua aineistoa käsiteltiin luottamuksellisesti. Kyselyn kysymykset muotoiltiin ohjelmoinnillisen ajattelun oppimisen teorian pohjalta (ks. Brennan & Resnick, 2012). Kysely pilotoitiin kaksi kertaa ennen sen julkaisua. Kumpikin pilottikysely auttoi tutkijoita huomaamaan kehityskohteita kyselyssä. Esimerkiksi kyselyn ensimmäisessä versiossa kysymykset oli muotoiltu siten, että niillä pyrittiin löytämään suoria vastauksia tutkimuskysymyksiin. Myöhemmin kysymykset muotoutuivat uusiksi siten, että niillä pyrittiin saamaan vastaajien omia kokemuksia ohjelmoinnin opetuksesta. Kyselylomakkeen avointen kysymysten yhteyteen myös liitettiin vastaamista helpottavia aputekstejä, joissa pyrittiin avaamaan mahdollisimman tarkasti vastaajalle, minkä tyyppisiä vastauksia häneltä toivotaan, kuitenkin jättäen vastauksiin tulkinnanvaraa siten, ettei vastaajaa ohjattu antamaan täysin tietyntyyppisiä vastauksia. Apukysymyksissä esitellyt aihepiiriin liittyvät termit saattoivat kuitenkin ohjata vastaajaa vastauksissaan jonkin verran tietyntyyppisiin vastauksiin tai käyttämään vastauksissaan kyseisiä apukysymyksissä mainittuja käsitteitä tai termejä. Avoimiin kysymyksiin vastaamisesta tehtiin myös pakollista. Haasteita kyselylomakkeessa on kuitenkin esimerkiksi vastaajien mahdollinen perehtymättömyys aihepiiriin, vastausväsymys pitkien avoimien kysymyksien edessä sekä se, ovatko vastaajat käsittäneet kysymyksen samalla tavoin kuin tutkijat. Lisäksi kyselylomakkeessa on muutamia eroavaisuuksia, kuten mahdottomuus syventää tai tarkistaa vastauksia kysymällä lisäkysymyksiä, esimerkiksi haastatteluna kerättyyn aineistoon verrattuna. Kyselytutkimus mahdollistaa kuitenkin vastaajalle mahdollisuuden pohtia ja tarkistaa vastauksiaan, sekä laajemman otannan tutkielman aikataulun huomioiden. (Hirsjärvi ym., 2010.)

Tutkielman aineistonkeruussa käytetyssä kyselyssä kysyttiin myös vastaajien näkemyksiä siiten, miten opettajien valmiuksia opettaa ohjelmointia voitaisiin kehittää. Kysymys ja siihen saadut vastaukset jätettiin kuitenkin lopulta huomioimatta tutkielmaa tehdessä johtuen sen irrallisuudesta muuhun aihepiiriin. Koulujen ja kuntien resurssipulan tiedetään vaikuttavan ohjelmoinnin opetuksen nykytilaan, vaikka sen vaikutukset eivät tässä tutkielmassa suoraa olekaan havaittavissa (Sentace & Czismadia, 2016). Vastaavasti ohjelmoinnin opetuksen käytänteitä kartoittavalla kysymyksellä saatiin laaja ja monialainen aineisto liittyen erilaisiin ohjelmoinnin opetukseen liittyviin työtapoihin, tehtäviin ja muihin käytänteisiin, joita olisi mahdollista tutkia erikseen tarkemmin.

Lähteet

Ala-Mutka, K. (2004). Problems in learning and teaching programming. Luettu 25.4.2021 https://www.cs.tut.fi/%7Eedge/literature_study.pdf

Balanskat, A., & Engelhardt, K. (2015). Computer Programming and Coding: Priorities, School Curricula and Initiatives Across Europe. European Schoolnet. Luettu 27.3. https://www.researchgate.net/publication/284139559_Computing_our_future_Computer_programming_and_coding_Priorities_school_curricula_and_initiatives_across_Europe

Bocconi, S., Chioccariello, A., Dettori, G., Ferrari, A., & Engelhardt, K. (2016). Developing Computational Thinking in Compulsory Education-Implications for policy and practice. Luxembourg: Publications Office of the European Union. Luettu 25.3. dx.doi.org/10.2791/792158

Bresnihan, N., Millwood, R., Oldham, E., Strong, G., & Wilson, D. (2015). A critique of the current trend to implement computing in schools. Pedagogika, 65(3), 292–300. Luettu 27.3. https://www.researchgate.net/publication/290993095_A_critique_of_the_current_trend_to_implement_computing_in_schools

Brennan, K. & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. Paper presented at the American Educational Research Association (AERA) meeting. Vancouver, BC, Canada. Luettu 2.3. https://web.media.mit.edu/~kbrennan/files/Brennan_Resnick_AERA2012_CT.pdf

Curzon, P., Black, J., Meagher, L. R. & McOwan, P. W. (2009) cs4fn.org: Enthusing students about Computer Science. Proceedings of Informatics Education Europe IV, 73–80. Luettu 4.3. https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.414.9456&rep=rep1&type=pdf

Denning, P. & Tedre, M. (2019). Computational Thinking. The MIT Press. Luettu 3.3. http://web.a.ebscohost.com.pc124152.oulu.fi:8080/ehost/ebookviewer/ebook/bmxlYmtfXzIxMDkxMjZfX0FO0?sid=f48ea1d4-675d-4654-9f40-85e71acd73b4@sessionmgr4006&vid=0&format=EB&rid=1

Denning, P. & Tedre, M. (2016). The Long Quest for Computational Thinking. Proceedings of the 16th Koli Calling Conference on Computing Education Research, November 24–27, Koli, Finland: pp. 120–129. Luettu 3.3. http://denninginstitute.com/pjd/PUBS/long-quest-ct.pdf

Dufva, T., & Dufva, M. (2016). Metaphors of code—Structuring and broadening the discussion on teaching children to code. Thinking Skills and Creativity, 22, 97-110. Luettu 10.3. https://doi.org/10.1016/j.tsc.2016.09.004

Eskola, J. & Suoranta, J. (2014). Johdatus laadulliseen tutkimukseen. 10. painos. Tampere: Vastapaino.

Euroopan komissio. (2010). Komission tiedonanto Euroopan parlamentille, neuvostolle, Euroopan talous- ja sosiaalikomitealle ja alueiden komitealle. Euroopan digitaalistrategia. Bryssel 19.5.2010. KOM(2010)245 lopullinen. Luettu 15.3. https://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=COM:2010:0245:FIN:FI:PDF

Falloon, G. (2015). Building computational thinking through programming in K-6 education: A New Zealand experience. Proceedings of EDULEARN15 Conference (pp. 0882–0892). Luettu 10.4. https://www.researchgate.net/publication/285590254_BUILDING_COMPUTATIONAL_THINKING_THROUGH_PROGRAMMING_IN_K-6_EDUCATION_A_NEW_ZEALAND_EXPERIENCE

Fidai, A., Capraro, M. M., & Capraro, R. M. (2020). "Scratch"-ing computational thinking with Arduino: A meta-analysis. Thinking Skills and Creativity, 38, 100726. Luettu 12.3. https://doi.org/10.1016/j.tsc.2020.100726

Grover, S., Jackiw, N. & Lundh, P. (2019). Concepts before coding: Non-programming interactives to advance learning of introductory programming concepts in middle school. Computer Science Education, 29(2–3), 106–135. Luettu 15.3. https://www.researchgate.net/publication/330916641_Concepts_before_coding_non-programming_interactives_to_advance_learning_of_introductory_programming_concepts_in_middle_school

Hirsjärvi, S., Remes, P. & Sajavaara, P. (2010). Tutki ja kirjoita. Helsinki. Tammi.

Hyvönen, M., Lappalainen, V. & Lakanen, A. J. (2012). Ohjelmointi 1: C. Luentomoniste/Jyväskylän yliopisto, tietotekniikan laitos, (17). Luettu 3.4. https://jyx.jyu.fi/bitstream/handle/123456789/47417/978-951-39-4859-7.pdf?sequence=1

Ihanainen, S. (2020). Ohjelmoinnin opetuksen näkymät alakouluissa. Pro gradu -tutkielma. University of Jyväskylä. Luettu 3.4. https://jyx.jyu.fi/handle/123456789/73440

Impagliazzo, J., Campbell-Kelly, M., Davies, G., Lee, J. A. N. & Williams, M. R. (1998). History in the Computing Curriculum. Luettu 11.4. dx.doi.org/10.1109/85.759364

International Society for Technology in Education [ISTE]. (2021). Luettu 1.4. https://www.iste.org/

Jalkanen, J., Järvenoja, M., & Litola, K. (2012). Muuttuva maailma, erilaisia oppijoita-millainen oppimisympäristö?. Äidinkielen opettajain liiton vuosikirja.

Kafai, Y. B. & Burke, Q. (2013). The social turn in K-12 programming: Moving from computational thinking to computational participation. Luettu 1.4. doi: 10.1145/2445196.2445373.

Kalelioglu, F. & Gülbahar, Y. (2014). The effects of teaching programming via Scratch on problem solving skills: a discussion from learners' perspective. Informatics in Education, 13(1), 33–50. Luettu 20.3. https://www.researchgate.net/publication/271746747_The_Effects_of_Teaching_Programming_via_Scratch_on_Problem_Solving_Skills_A_Discussion_from_Learners%27_Perspective

Kangas, J., & Vartiainen, J. (2019). Ohjelmoinnin ABC varhaiskasvatuksessa.

Kekäläinen, O. (2015). Onko automatisointiajattelu paras suomennos käsitteestä "computational thinking?" Teoksessa Viteli, J. & Östman, A. (toim.) Tuovi 13: Interaktiivinen tekniikka koulutuksessa 2015-konferenssin tutkijatapaamisen artikkelit (27–29). Tampereen yliopisto.

Kert, S. B., Erkoç, M. F., & Yeni, S. (2020). The effect of robotics on six graders' academic achievement, computational thinking skills and conceptual knowledge levels. Thinking Skills and Creativity, 38, 100714. Luettu 15.3. https://doi.org/10.1016/j.tsc.2020.100714

Klopfer, E., & Yoon, S. (2005). Developing games and simulations for today and tomorrow's tech savvy youth. TechTrends, 49(3), 33–41. Luettu 29.3. https://www.researchgate.net/publication/238113555_Developing_games_and_simulations_for_today_and_tomorrow%27s_tech_savvy_youth

Kurkinen, H. (2018). " Koodaaminen ei oo sitä tietokonenippeliä pelkästään": luokanopettajien käsityksiä ja kokemuksia ohjelmoinnin opettamisesta.

Laine, M., Bamberg, J., & Jokinen, P. (2007). Tapaustutkimuksen taito. Luettu 17.4. https://researchportal.tuni.fi/en/publications/tapaustutkimuksen-taito

Lye, S. Y. & Koh, J. H. L. (2014). Review on teaching and learning of computational thinknig through programming: What is next for K-12? Computers in Human Behavior 41, 51–61. Luettu 14.3. https://www.sciencedirect.com/science/article/pii/S0747563214004634

Madigan, D. & Martin, A. (2006). Digital Literacies for Learning. London. Facet Publishing.

Majaranta, P. (2002). Ohjelmointi on helppoa –lapsikin sen osaa. Teoksessa P. Hietala & S. Ovaska(toim.) Lasten käyttöliittymät. Tampere. Tampereen yliopisto.

Makkonen, J., & Pyykönen, A. (2018). " Se on mun mielest taas yks tapa rikastuttaa sitä opiskelua ja oppimista, itekki oppii sit uusii juttui.": alakoulun opettajien käsityksiä ohjelmoinnin opettamisesta. University of Jyväskylä. Luettu 20.3. https://jyx.jyu.fi/handle/123456789/59407

Malan, D. J., & Leitner, H. H. (2007). Scratch for budding computer scientists. ACM SIGCSE Bulletin, 39(1), 223–227. Luettu 19.3. https://www.researchgate.net/publication/38413358_Scratch_for_Budding_Computer_Scientists

Maloney, J., Resnick, M., Rusk, N., Silverman, B., & Eastmond, E. (2010). The scratch programming language and environment. ACM Transactions on Computing Education (TOCE), 10(4), 1–15. Luettu 24.3. https://web.media.mit.edu/~jmaloney/papers/Scratch-LangAndEnvironment.pdf

Mason, S. L., & Rich, P. J. (2019). Preparing elementary school teachers to teach computing, coding, and computational thinking. Contemporary Issues in Technology and Teacher Education, 19(4), 790-824. Luettu 7.3. https://citejournal.org/volume-19/issue-4-19/general/preparing-elementary-school-teachers-to-teach-computing-coding-and-computational-thinking/

McClelland, K. & Grata, L. (2018). A Review of the Importance of Computational Thinking in K-12. Instructional Technology and Leadership. Duquesne University. Pittsburgh, PA, USA.

McCormack, A. (2014). The e-Skills Manifesto. A Call to Arms. Brussels.

Mertala, P., Palsa, L., & Dufva, T. S. (2020). Monilukutaito koodin purkajana: Ehdotus laaja-alaiseksi ohjelmoinnin pedagogiikaksi. Media & viestintä, 43(1). Luettu 5.4. DOI: https://doi.org/10.23983/mv.91079

Mohagegh, M. & McCauley M. (2016). Computational Thinking: The Skill Set of the 21st Century International Journal of Computer Science and Information Technologies, Vol. 7 (3), 2016, 1524–1530. Luettu 22.4. http://ijcsit.com/docs/Volume%207/vol7issue3/ijcsit20160703104.pdf

Moilanen, P. (2018). Kynnyskäsitteet ohjelmoinnin oppimisessa. Helsingin yliopisto

Mouza, C., Pan, Y. C., Yang, H. & Pollock, L. (2020). A Multiyear Investigation of Student Computational Thinking Concepts, Practices, and Perspectives in an After-School Computing Program. Journal of Educational Computing Research. 2020, Vol. 58(5). Luettu 4.5. https://doi.org/10.1177/0735633120905605

Mykkänen, J., & Liukas, L. (2014). Koodi 2016–Ensiapua ohjelmoinnin opettamiseen peruskoulussa. Helsinki. Luettu 3.3.: http://koodi2016.fi/lataa.html

Naharro-Berrocal, F., Pareja-Flores, C., Urquiza-Fuentes, J., & Velázquez-Iturbide, J. á. (2002). Approaches to comprehension-preserving graphical reduction of program visualizations. Proceedings of the 2002 ACM symposium on Applied computing-SAC '02 (pp. 771). Madrid, Spain: ACM Press. Luettu 14.4: https://doi.org/10.1145/508791.508941

National Research Council. (2013). Frontiers in Massice Data Analysis. Washington, DC: The National Academies Press. Luettu 19.4. https://doi.org/10.17226/18374

Opetushallitus. (2021). Uudet lukutaidot. Luettu 17.4. https://uudetlukutaidot.fi/

Opetushallitus. (2014). Perusopetuksen opetussuunnitelman perusteet 2014. Helsinki.

Opetus- ja kulttuuriministeriö. (2013). Opetus- ja kulttuuriministeriön älystrategia: OKM-KIDE. Luettu 19.4. https://julkaisut.valtioneuvosto.fi/handle/10024/75282

Otterborn, Schönborn, Hulten 2019. Investigating Preschool Educators' Implementation of Computer Programming in Their Teaching Practice. Luettu 15.4. https://link.springer.com/article/10.1007/s10643-019-00976-y

Quigley, C. F., & Herro, D. (2016). "Finding the joy in the unknown": Implementation of STEAM teaching practices in middle school science and math classrooms. Journal of Science Education and Technology, 25(3), 410–426. Luettu 15.5. DOI:10.1007/s10956-016-9602-z

Raivonen, P. (2018). Rohkeasti vain kokeilemaan. Tutkimus alakoulun opettajien valmiudesta opettaa ohjelmointia. Luettu 23.4. https://helda. helsinki. fi/bitstream/handle/10138/233147/PetraRaivonen_ Gradu. pdf

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., ... & Kafai, Y. (2009). Scratch: programming for all. Communications of the ACM, 52(11), 60-67. Luettu 19.4. https://doi.org/10.1145/1592761.1592779

Saarikoski, P. (2006). Koneen ja koulun ensikohtaaminen. Tekniikan Waiheita, 24(3), 5–19. Luettu 19.4. https://journal.fi/tekniikanwaiheita/article/view/63817/25125

Sáez-López, J. M., Román-González, M., & Vázquez-Cano, E. (2016). Visual programming languages integrated across the curriculum in elementary school: A two year case study using “scratch” in five schools. Computer & Education, 97, 129–141. Luettu 17.4. https://www.sciencedirect.com/science/article/pii/S0360131516300549

Schwartz, J., Stagner, J. & Morrison, W. (2006). Kid’s programming language (Kpl). ACM SIGGRAPH 2006 Educators program on-SIGGRAPH ’06 (pp. 52). Boston, Massachusetts: ACM Press. Luettu 5.4. https://doi.org/10.1145/1179295.1179348

Segredo, E., Miranda, G., León, C. & Santos, A. (2016). Developing computational thinking abilities instead of digital literacy in primary and secondary school students. In Smart Education and e-Learning 2016 (pp. 235–245). Springer, Charm. Luettu 17.4. https://www.researchgate.net/publication/303761211_Developing_Computational_Thinking_Abilities_Instead_of_Digital_Literacy_in_Primary_and_Secondary_School_Students)

Sentace, S., Csizmadia, A. (2016). Computing in the curriculum: Challenges and strategies from a teacher’s perspective. Luettu 20.5. doi:10.1007/s10639-016-9482-0

Stolee, K. T., & Fristoe, T. (2011). Expressing computer science concepts through Kodu game lab. InProceedings of the 42nd ACM technical symposium on Computer science education - SIGCSE ’11 (pp. 99). Dallas, TX, USA: ACM Press. Luettu 5.4. https://doi.org/10.1145/1953163.1953197

Taina, J. (2015). Peruskoulun ohjelmointiopetus. Matematiikkalehti Solmu. Luettu 6.3. https://matematiikkalehtisolmu.fi/2015/2/ohjelmointiopetus.pdf

Tanhua-Piiroinen, E., Kaarakainen, S. S., Kaarakainen, M. T., Viteli, J., Syvänen, A., & Kivinen, A. (2019). Digiajan peruskoulu.

Tuomi, J. & Sarajärvi, A. (2018). Laadullinen tutkimus ja sisällönanalyysi. Tammi.

Trilling, B. & Fadel, C. (2009). 21st century skills: Learning for life our times. John Wiley & Sons.

Tsai, M-J., Wang, C-Y. & Hsu, P-F. (2019). Developing the Computer Programming Self-Efficacy Scale for Computer Literacy Education. Journal of Educational Computing Research, 56. Luettu 16.4. https://www.sciencedirect.com/science/article/pii/S0360131520302219)

Vuorikari, R., Punie, Y., Carretero, S. & Van den Brande, L. (2016). DigComp 2.0: The Digital Competence Framework for Citizens. Update Phase 1: The Conceptual Reference Model. European Commission. Joint Research Centre. Luettu 14.4. dx.doi.org/10.2791/11517

Vuorinen, J. (2019). Opettajien täydennyskoulutus, ohjelmointi ja ohjelmoinnillinen ajattelu. Teoksessa Portaankorva-Koivisto, P., Heinonen, M., & Mäkelä, E. (toim.). Kuka meitä opettaa? Esseitä tietotekniikan opetuksesta (91–115).

Wei, X., Lin, L., Meng, N., Tan, W., Kong, S-C., Kinshuk. (2021). The effectiveness of partial pair programming on elementary school students' Computational Thinking skills and self-effiacy. Computer & Education, Volume 160.

Wilson, A., & Moffat, D.C. (2010). Evaluating Scratch to introduce younger school children to programming. Luettu 11.5. http://scratched.gse.harvard.edu/sites/default/files/wilson-moffat-ppig2010-final.pdf

The White House. (2016). Computer science for all. Luettu 19.4. http://bit.ly/2tcPrAj

Wing, J. (2006). Computational Thinking. Communications of the ACM, Volume 49. Luettu 17.4. https://dl.acm.org/doi/10.1145/1118178.1118215

Wu, L., Looi, C. K., Multisilta, J., How, M. L., Choi, H., Hsu, T. C., & Tuomi, P. (2020). Teacher's perceptions and readiness to teach coding skills: A comparative study between Finland, mainland China, Singapore, Taiwan, and South Korea. The Asia-Pacific Education Researcher, 29(1), 21-34.

Wu, W. Y., Chang, C. K., & He, Y. Y. (2010). Using Scratch as game-based learning tool to reduce learning anxiety in programming course. Proceedings of Global Learn Asia Pacific 2010-Global Conference on Learning and Technology (pp. 1845-1852). Penang, Malaysia: Association for the Advancement of Computing in Education (AACE).

Yadav, A., Good, J., Voogt, J. & Fisser, P. (2017) Computational Thinking as and Emerging Competence Domain. Competence-based Vocational and Professional Education, Volume 23, pp. 1051–1067). Luettu 9.4. https://doi.org/10.1007/978-3-319-41713-4_49

Yucel, Ö., Karahoca, D. & Karahoca, A. (2016). The effects of problem-based learning on cognitive flexibility, self-regulation skills and students' achievements. Global Journal of Information Technology: Emerging Technologies, 6(1). Luettu 19.4. dx.doi.org/10.18844/gjit.v6i1.394

Özmen, B. & Altun, A. (2014). "Undergraduate students' experiences in programming: difficulties and obstacles." Turkish Online Journal of Qualitative Inquiry, 5(3), 1–27.Luettu 4.4. DOI:10.17569/tojqi.20328

Kuviot

Taulukot

Liite 1 / Tutkimuskysely

Ohjelmoinnin opettaminen

Pakolliset kentät merkitään asteriskilla (*) ja ne tulee täyttää lomakkeen viimeistelemiseksi.

Hyvä ohjelmoinnin opettamisesta kiinnostunut opettaja!

Olemme kaksi luokanopettajaopiskelijaa Oulun yliopistosta. Teemme pro gradu -tutkielmaa, jonka tarkoituksena on selvittää opettajien kokemuksia ja näkemyksiä ohjelmoinnin opettamisesta.

Toivomme vastauksia perusopetuksen (0-9 luokka-asteen) opettajilta.

Pro gradu -tutkielman aineiston keräämme tällä sähköisellä lomakkeella ja mahdollisesti myöhemmin toteutettavilla haastatteluilla. Vastaamalla tähän lomakkeeseen annat suostumuksesi aineiston käyttöön tässä tutkielmassa. Aineistoa käsitellään luottamuksellisesti. Tutkimukseen on mahdollista osallistua anonyymisti.

Lomake koostuu taustatietojen kartoituksesta, sekä sen jälkeisestä kuudesta avoimesta kysymyksestä.

Lomakkeeseen vastaaminen vie aikaa keskimäärin n. 15-20 minuuttia.
Tutkimukseen on aikaa osallistua 11.4.2021 klo 23:59 asti.
Suosittelemme vastaamista tietokoneella.

Mikäli olet halukas osallistumaan myöhemmin mahdollisesti toteutettaviin haastatteluihin, voit jättää yhteystietosi lomakkeen lopussa.

Jokainen vastaus on tärkeä, kiitos osallistumisestasi!
Kerromme mielellämme lisää tutkimuksestamme sähköpostitse.
jaakko.korpela@student.oulu.fi tai samuel.koski@student.oulu.fi

1. Nykyinen työtehtäväsi *

- ○ Luokanopettaja 0.-2. lk
- ○ Luokanopettaja 3.-4. lk
- ○ Luokanopettaja 5.-6. lk
- ○ Aineenopettaja (mikä/mitkä oppiaineet?)
- ○ Muu, mikä?

2. Kuinka kauan olet työskennellyt opetusalalla? *

- ○ alle 1 vuosi
- ○ 1-4 vuotta
- ○ 5-9 vuotta
- ○ 10-19 vuotta
- ○ yli 20 vuotta

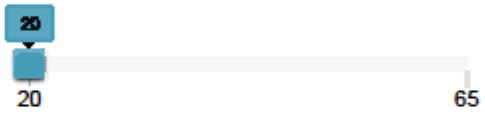
3. Kuinka pitkään olet toiminut nykyisessä työtehtävässäsi? *

- ○ alle 1 vuotta
- ○ 1-4 vuotta
- ○ 5-9 vuotta
- ○ yli 10 vuotta

4. Onko sinulla ohjelmoinnin opetuksessa hyödyksi katsottavia erikoistumisopintoja? Jos on, erittele lyhyesti mitä opintoja sinulla on *(esimerkiksi sivuaine, jokin opintokokonaisuus tai lisäkoulutus).*

Jos ei ole, vastaa "ei". *

5. Ikäsi *



6. Missä maakunnassa työskentelet tällä hetkellä?

- ○ Ahvenanmaa
- ○ Etelä-Karjala
- ○ Etelä-Pohjanmaa
- ○ Etelä-Savo
- ○ Kainuu
- ○ Kanta-Häme
- ○ Keski-Pohjanmaa
- ○ Keski-Suomi
- ○ Kymenlaakso
- ○ Lappi
- ○ Pirkanmaa
- ○ Pohjanmaa
- ○ Pohjois-Karjala
- ○ Pohjois-Pohjanmaa
- ○ Pohjois-Savo
- ○ Päijät-Häme
- ○ Satakunta
- ○ Uusimaa
- ○ Varsinais-Suomi

7. Arvioi, kuinka hyvin osaat opettaa ohjelmointia peruskoulutasolla *

8. Mitä ohjelmointiin liittyviä käsitteitä pidät tärkeimpinä ohjelmoinnin opetuksessa? *

Mainitse enintään viisi mielestäsi tärkeintä käsitettä.

9. Millaisten tehtävien avulla olet opettanut ohjelmointia? *

Kerro esimerkkejä tehtävistä, joita olet käyttänyt ohjelmoinnin opetuksessa. Perustele lyhyesti, miksi olet käyttänyt kyseisiä tehtäviä tai tehtävätyyppejä.

Miten ohjelmoinnin opetukselle ominaiset tehtävätyypit, kuten *testaaminen, debuggaaminen (virheenkorjaaminen), remiksaaminen (uudelleenkäyttäminen)* ja niin edelleen, näkyvät käyttämissäsi tehtävissä?

10. Millaisia työtapoja olet käyttänyt opettaessasi ohjelmointia? *

Kerro esimerkkejä työtavoista, joita olet käyttänyt ohjelmoinnin opetuksessa. Perustele lyhyesti, miksi olet käyttänyt mainitsemiasi työtapoja.

11. Mitkä ovat mielestäsi keskeiset tavoitteet, joiden vuoksi ohjelmoinnin opettaminen peruskoulussa on tärkeää? *

12. Mitä vaikutuksia ohjelmointiosaamisella on oppilaiden tulevaisuudelle? *

13. Arvioi, kuinka hyvin oman koulusi muut opettajat osaavat opettaa ohjelmointia peruskoulutasolla? *

14. Pohdi, kuinka opettajien valmiuksia opettaa ohjelmointia voitaisiin kehittää. *

Mainitse enintään kolme keinoa kehittää opettajien valmiuksia opettaa ohjelmointia.

15. Jatkotutkimukseen osallistuminen

Jos haluat osallistua mahdolliseen haastatteluna toteutettavaan jatkotutkimukseen, jätä yhteystietosi alapuolella oleviin kenttiin. Jatkotutkimukseen osallistuminen ei ole pakollista.

Mikäli tulet valituksi jatkotutkimukseen, lähestymme sinua sähköpostitse.

Etunimi

Sukunimi

Sähköposti

Liite 2 / Kyselylinkin yhteyteen liitetyt motivoivat kuvat